



**POLITECNICO**  
MILANO 1863

MSC IN COMPUTER SCIENCE AND ENGINEERING

SOFTWARE ENGINEERING 2 PROJECT

---

## **TrackMe Requirement Analysis and Specification Document**

---

Professor:  
Elisabetta di Nitto

Authors :  
Andrea Biscontini - 901310  
Marco Gelli - 901470  
Alvise de'Faveri Tron - 920882

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Purpose	4
1.2	Scope	4
1.2.1	Description of the given problem	4
1.2.2	Goals	5
1.3	Definitions, Acronyms, Abbreviations	5
1.3.1	Definitions	5
1.3.2	Acronyms	5
1.3.3	Abbreviations	6
1.4	Revision history	6
1.5	Document Structure	6
<b>2</b>	<b>Overall Description</b>	<b>7</b>
2.1	Product perspective	7
2.1.1	Data4Help	8
2.1.2	AutomatedSOS	9
2.1.3	Track4Run	10
2.2	Product functions	11
2.2.1	Data4Help - User Data Acquisition	11
2.2.2	Data4Help - User Data Sharing	11
2.2.3	AutomatedSOS - Health Emergency Monitoring	11
2.2.4	Track4Run - Run Events Management	11
2.3	User characteristics	12
2.3.1	Actors	12
2.4	Assumptions, dependencies and constraints	13
2.4.1	Domain Assumptions	13
2.4.2	Software dependencies	13
2.4.3	Hardware constraints	14
<b>3</b>	<b>Specific Requirements</b>	<b>15</b>
3.1	External Interface Requirements	15
3.1.1	User Interface	15
3.1.2	Hardware Interfaces	20
3.1.3	Software Interfaces	20
3.1.4	Communication Interfaces	20
3.2	Functional Requirements	20
3.2.1	Requirements	20
3.2.2	Scenarios	23
3.2.3	Use Cases	25
3.2.4	Sequence Diagrams	35
3.3	Performance Requirements	47
3.4	Design Constraints	47
3.4.1	Regulatory policies	47
3.4.2	Hardware limitations	47
3.5	Software System Attributes	47
3.5.1	Reliability	47
3.5.2	Availability	47
3.5.3	Security	47
3.5.4	Maintainability	48

3.5.5	Portability . . . . .	48
<b>4</b>	<b>Formal Analysis Using Alloy . . . . .</b>	<b>49</b>
4.1	World Generated . . . . .	55
4.2	Proof of consistency . . . . .	56
<b>5</b>	<b>Effort Spent . . . . .</b>	<b>57</b>
5.1	Andrea Biscontini . . . . .	57
5.2	Alvise de' Faveri Tron . . . . .	57
5.3	Marco Gelli . . . . .	57
<b>6</b>	<b>Reference . . . . .</b>	<b>58</b>
6.1	Reference . . . . .	58
6.2	Tools Used . . . . .	58

# 1 Introduction

## 1.1 Purpose

This document is the Requirement Analysis and Specification Document of the Data4Help system and the two subsystems working on top of it: AutomatedSOS and Track4Run. In this document we will describe the general purpose of the system, the main goals that it has to achieve, the functional and non-functional requirements that have to be met and the domain assumptions and constraints that have been identified.

This document is addressed to all the stakeholders of the system: customers (end users or third parties), system analysts, project managers and developers. Here a brief overview of the product will be given.

Data management is a very popular topic nowadays. It has applications in multiple fields and it may result in many benefits when properly employed: for example medical research, healthcare and the fitness world have greatly benefited from an extensive use of personal data in the past years. On the other hand, data treatment and security has been an increasingly important concern among our society, which resulted in a more strict and aware regulatory policies.

Data4Help is a system designed to address the issue of collecting user data from different sources like smart devices or external applications and ensure a secure and reliable way to share them with interested companies. Additionally, some other functionalities are added to the system by AutomatedSOS and Track4Run.

The main purpose of AutomatedSOS is to offer to Data4Help users an automated solution to monitor their health status and call an ambulance that will rescue them in case of emergency. This is done by exploiting the data collected by the Data4Help system.

Finally, Track4Run has the objective to solve the organizational problems that arise in the management of sport events, in particular running events that in our days are becoming more and more popular, even among amateurs. In fact, this system gives to each user the possibility to organize and participate to runs, providing also the possibility, for those who don't want to run, to watch the participants from the sideline.

## 1.2 Scope

### 1.2.1 Description of the given problem

As stated in the above section, Data4Help main goal is to collect user data and make them available to third parties, all while guaranteeing the user privacy and consensus in personal data processing.

To collect these data, the system needs to connect to users' smart devices and wearables so that they can send the retrieved information to the TrackMe proprietary system. This data are then processed by Data4Help and made available for third party organizations under certain conditions.

In particular, we can divide requests in two different types: single user data requests, that are forwarded directly to the individual, who can accept or refuse them, and data requests for groups of individuals, that are handled by Data4Help and are carried out only if there is the possibility to properly anonymize the data. Third parties could also desire to look for future changes in the data they requested: for this reason an auxiliary subscription to new data is also provided.

Moving to AutomatedSOS, it can be used to help elderly or sick people to monitor their health conditions and intervene in the case of an emergency. In fact, the goal of AutomatedSOS is to be very reactive (maximum 5 seconds) in detecting possible health problems and immediately call an ambulance to the user location.

Instead, Track4Run is designed for being used by a multitude of different users: any user can become the organizer of a run by creating one. The run creation procedure is made really simple: the user just needs to insert the information related to the event and select a route for the run on the map. When a run

is created, every other user can enroll to it. To give an even better service, there is also the possibility to follow every runner's position on a live GPS map.

### 1.2.2 Goals

- **[G1]** Data4Help must be able to keep track of real time health status and position from registered users
- **[G2]** Data4Help should allow third parties to gather information from a single user or from an anonymous group of users
- **[G3]** Data4Help should allow third parties to subscribe to new data and receive them as soon as they're produced
- **[G4]** AutomatedSOS should be able to identify an health emergency when the user data are below/exceeding a certain thresholds
- **[G5]** AutomatedSOS must call an ambulance when it detects a health emergency
- **[G7]** Track4Run must allow a user to create and manage running events
- **[G8]** Track4Run must allow a user to participate to an organized run
- **[G9]** Track4Run must allow a spectator to track the position of the participants of an ongoing run

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Definitions

- **Third Party:** Any system that is not Data4Help that wants to access to Data4Help user's data
- **Data Request:** The action that third parties can make to retrieve users data
- **Notification:** An email, SMS or any other way in which a user can be alerted of any event of the system
- **Data Source:** An external device or application that can collect data from the user
- **Synchronization:** The procedure to connect a Data Source with the Data4Help system
- **Parameter:** A generic physical or health information regarding the user that can be collected from a device
- **Threshold:** A value defining when a certain parameter is out of the normal range
- **Emergency Status:** A status in which one or more parameters of an AutomatedSOS user exceeded their thresholds

### 1.3.2 Acronyms

- **RASD:** Requirement Analysis and Specification Document
- **API:** Application Programming Interface
- **GPS:** Global Positioning System
- **GDPR:** General Data Protection Regulation

### 1.3.3 Abbreviations

- **Gn:** n-th goal
- **Dn:** n-th domain assumption
- **Rn:** n-th functional requirement

## 1.4 Revision history

Date	Applied Changes
11/11/2018	First issue of the document
13/11/2018	Added Sequence Diagrams images in the Functional Requirements section. Added Mockup images in the External Interfaces section. Added Alloy text and images in the Alloy section. Fixed use case tables and corresponding order. Fixed numbers of Requirements and Domain Assumptions in the Functional Requirements section. Added Revision History in the introduction.
10/12/2018	Removed R18 as it is not needed for the system to function. Also removed third party mockups.
16/12/2018	Modified Alloy model, Scenarios and Use Cases in line with design document.
17/12/2018	Brief References and Tools Used section added

## 1.5 Document Structure

Chapter 1 gives an introduction to the problem and describes the goals and purpose of the application.

Chapter 2 presents an overall description of the system. Firstly a description of the domain model and shared phenomena are provided coupled with detailed UML diagrams. Secondly the major functions of the system are more precisely specified and connected with the goals and domain constraints of the system.

Chapter 3 contains all the requirements that the system must meet in order to accomplish the given goals. In particular mockups of the user interfaces, use cases and sequence diagrams are presented to give a complete description of what features the system should and should not offer.

Chapter 4 includes the Alloy model with an example of a generated world.

Chapter 5 shows the effort by each group member.

## 2 Overall Description

### 2.1 Product perspective

Our product is an ecosystem of applications designed from scratch to accomplish different purposes.

In particular, the system is designed to be a set of three subsystems interacting with each other, as shown in the following diagram:

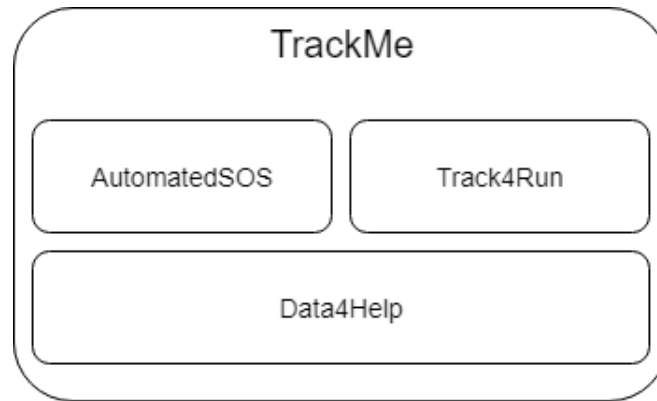


Figure 1: TrackMe ecosystem

Data4Help is the underlying backbone of our system that collects data from end users and makes it available to the third party applications.

The other two subsystems are built on top of Data4Help and provide to the users different services: AutomatedSOS offers an automated way of calling an ambulance in case of emergency, while Track4Run gives to the end users a platform to organize and participate to run events.

The following subsections give a more detailed analysis of the domain of each subsystem and are accompanied with the corresponding class diagrams. The aim of these diagrams is to identify the main actors and components of the subsystems and the relationship between them.

### 2.1.1 Data4Help

Data4Help is meant to be a stand-alone system that works as a gateway between user data *producers* and the third parties, which are data *consumers*. The central actor of this system is the user.

Each user can configure multiple data sources: each of them can collect different parameters that together form the user data. On the other hand, third parties can make some data requests, that can be targeted to a single user data or a group of data.

These requests can be of two different types: subscriptions, which will cause the third party to be updated whenever the chosen data set changes, or single requests.

Bearing these considerations in mind, the following diagram is given to better describe the domain of the application.

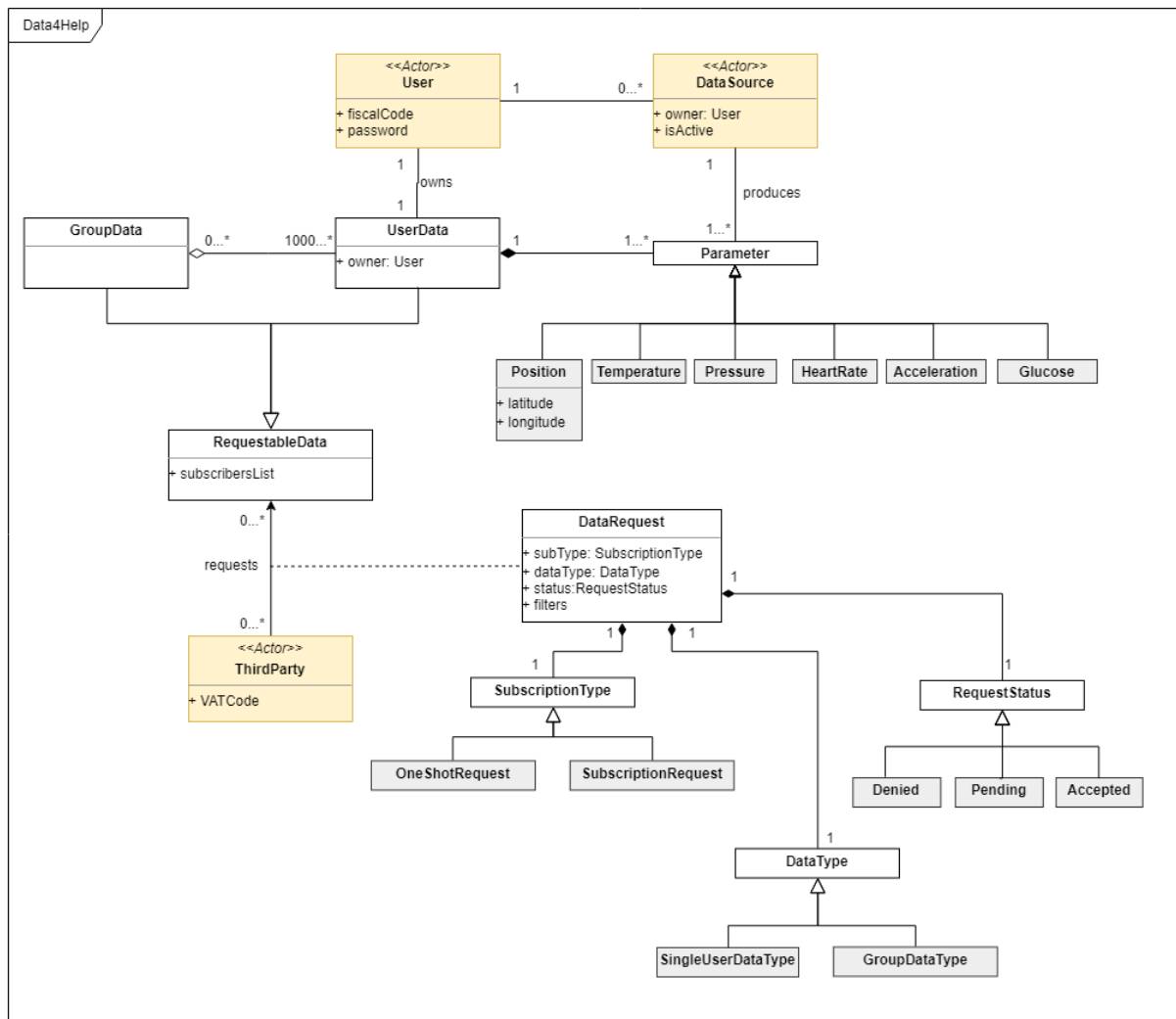


Figure 2: Data4Help Class Diagram

Please note that the parameter types listed here are the most commonly used in many kind of applications, but this doesn't exclude that more parameter types can be added in future.



### 2.1.2 AutomatedSOS

Differently from the previous one, AutomatedSOS is an application which relies on other systems, such as an Ambulance Calling System and Data4Help, to provide a specific service to the customers. The offered service is a monitoring system for Data4Help users, in which some threshold values can be set for each vital parameter. If one of these parameters exceeds his thresholds, this will cause the system to send the GPS location of the user in danger to an ambulance.

In this perspective, the AutomatedSOS system can be seen as one of the third parties of the previous diagram: for each new user, it makes a subscription request for the data of that user to the Data4Help system

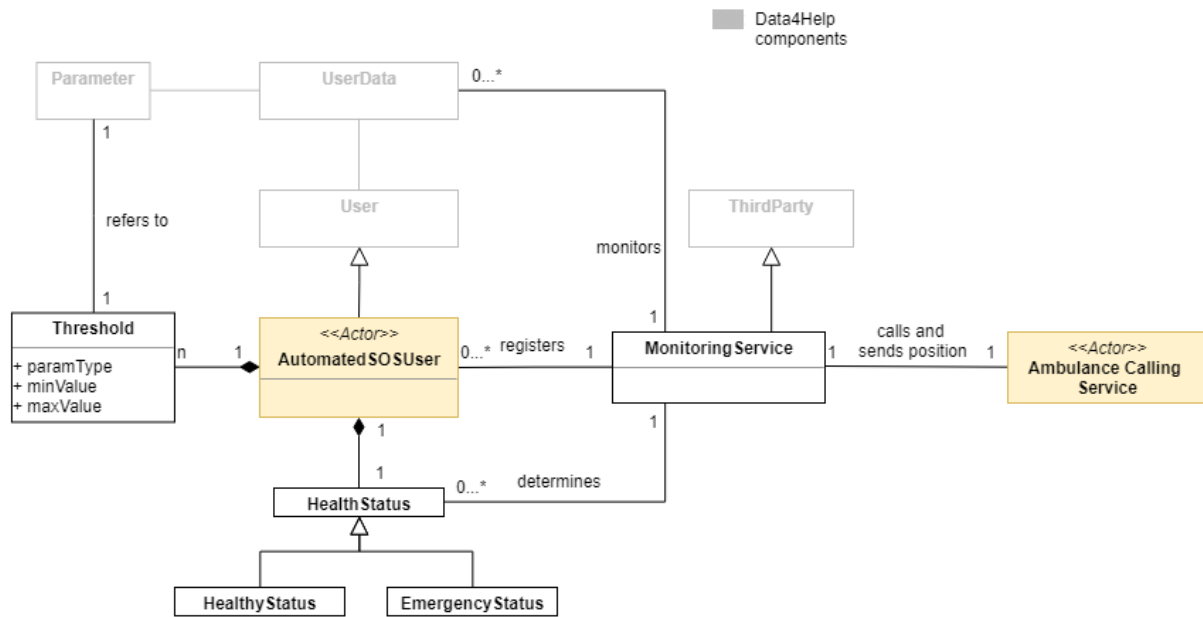


Figure 3: AutomatedSOS Class Diagram

### 2.1.3 Track4Run

Like AutomatedSOS, Track4Run is an application that uses Data4Help to monitor some of its users, in particular the runners who participate to run event. These run events are created and managed by organizers. On the other hand, spectators can see the position of the runners by using the dedicated service.

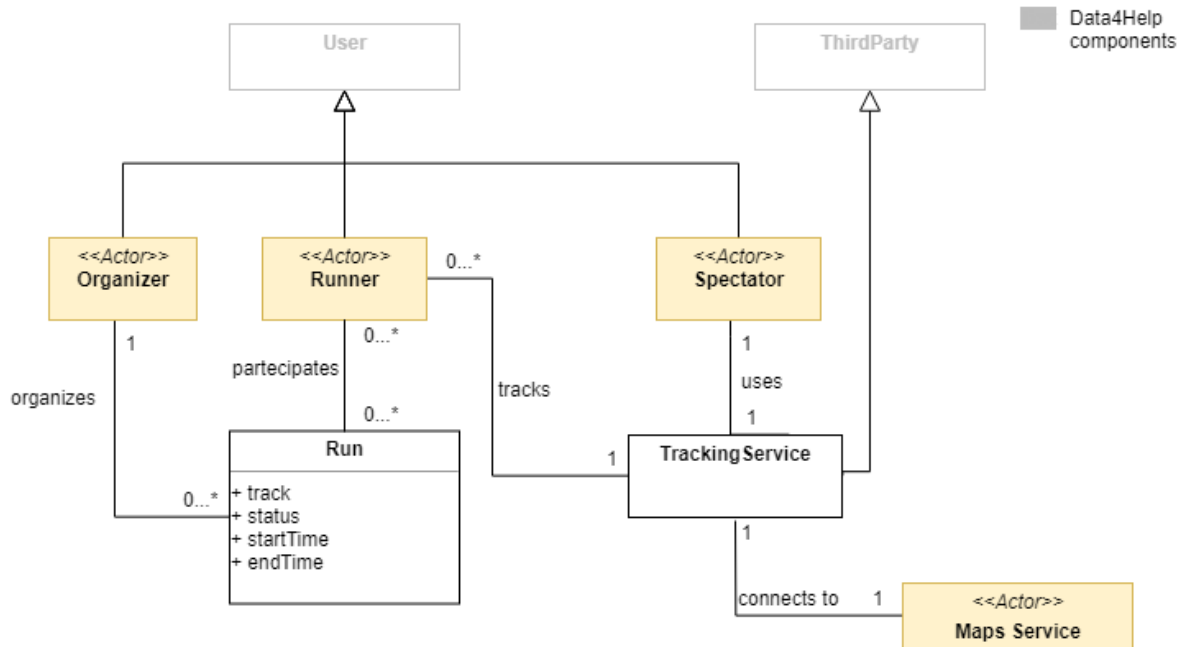


Figure 4: Track4Run Class Diagram

## **2.2 Product functions**

Considering all the goals this system has to accomplish, an additional and more detailed description is here provided for the main functions that each subsystem has to accomplish.

### **2.2.1 Data4Help - User Data Acquisition**

Data acquisition is the focus of Data4Help and it is also fundamental for all the subsystems built on top of it.

In order to provide this feature, the system must give to the user the possibility to store his position and state of health and share them with the interested third parties.

This kind of data can be acquired from different data sources, such as external applications, smart-phones, smartwatches or other similar devices. In order to collect the data from them, the user must firstly synchronize the device or application with Data4Help by providing his credentials and accepting the new data source.

The data source will then be able to retrieve real-time data from the user and send them to the Data4Help system, which in turn makes them available to the authorized third parties.

In any time the user must be able to add a new data source, decide which parameter can be monitored from which device, add a new data source or stop an existing one from sending new data.

### **2.2.2 Data4Help - User Data Sharing**

The system gives the possibility to third parties to register to the service and request users' data.

Data requests can be done by using an API which will be provided in order to encourage third parties to develop their own applications integrated with Data4Help services.

There are two different types of requested data: single users' data or multiple users' data. The first ones are forwarded to the user and need to be explicitly accepted by him before being carried out. On the other hand, multiple users' data requests do not need user approval, but they can be completed only if the number of users satisfying the search filters are enough to guarantee a proper data anonymization.

In order to be continuously updated, third parties can also make data subscription requests. In this case, a callback interface must be provided in order to be notified as soon as the monitored data changes.

Also, in this case, the user can at any time decide to revoke to the third parties the permission to access his data.

### **2.2.3 AutomatedSOS - Health Emergency Monitoring**

AutomatedSOS continuously monitors the health parameters selected by the user, exploiting the Data4Help system's data. The user of this system is provided with the possibility to set some thresholds, that can be default or changed at any moment.

As soon as one of them exceeds its pre-defined threshold, the system must contact an external interface that sends to an ambulance the position of the user in danger.

Between the emergency detection and the ambulance call, the user is notified that a threshold has been exceeded and a small window of time is granted to the user to abort the operation, so to minimize false-positive situations.

### **2.2.4 Track4Run - Run Events Management**

Run events are the central focus of the Track4Run subsystem. Its main goal is to provide the users with the possibility of organizing, participating and watching runs.

Each user will have the possibility to see all the runs that he has created and to create a new one. Creating a run means deciding a date and hour for starting and ending and select a track from a map: for this, the access to an external service of map visualization like Google Maps will be necessary. This run

can then be later deleted or postponed, or the track can be changed, and on each of these changes, the participant runners shall be informed.

The system also offers the possibility to search for ongoing runs and other planned runs that can be joined. In the first case, the user can decide a run to monitor and a map will appear with the live position of all the runners that are involved in that event. In the second case, joining a run means that somebody can track your position while you are running and you will receive notifications if some details change before it starts.

The position tracking is made using Data4Help, so the user's corresponding credentials are needed in order to use this application.

## 2.3 User characteristics

### 2.3.1 Actors

In the following section, a more specific characterization of the users and actors of each system is provided, based on the domain description that was previously offered.

- **Data4Help User:** in the Data4Help system the user is only a provider of information. After registration, he can access the platform's functionalities by logging in the system.
- **Third Party:** any organization or software that wants to access the user's data. This is the main direct user of the Data4Help services. He can access the system by registering with a trusted authentication method and using the token generated by his registration. AutomatedSOS and Track4Run are considered to be third parties according to this definition, even if they don't belong to an external organization.
- **Data Source:** any application or device that can be configured by the user to synchronize itself with the system, meaning that it can be authenticated using the user's Data4Help credentials. We can imagine that the existing companies in the wearable and smart devices world can make a deal with TrackMe to enable the users of their systems to add a Data4Help account, therefore synchronizing the device/application.
- **AutomatedSOS User:** is imagined to be a Data4Help user who wants to carefully monitor his health status, such as an elder user or someone who is undergoing or recovering from an illness.
- **Ambulance Calling System:** a supposedly existing external system that is able to provide an interface to access the national health-care system, used by AutomatedSOS.
- **Run Organizer:** a user who creates a run event. He must be a Data4Help user who has downloaded the Track4Run application.
- **Runner:** a user who participates to an organized run. Like the previous one, he has to be already registered to Data4Help in order to use the application.
- **Run Spectator:** a Data4Help user that accesses the Track4Run application to follow an ongoing run event.
- **Map Application:** an external service which provides a reliable visualization and tracking service of the GPS position on a map. An example could be *Google Maps*. This must be accessed from the Track4Run application through some kind of interface.
- **System Administrator:** someone who is in charge of administrating and maintaining the system. His credentials are generated by the application itself before the deployment, hence he doesn't have to register.

## **2.4 Assumptions, dependencies and constraints**

### **2.4.1 Domain Assumptions**

- [D1] Fiscal code uniquely identifies a user of the system.
- [D2] Every user owns at least one device capable of retrieving correct real-time health parameters and location
- [D3] User devices must grant to the system access to the requested data
- [D4] User devices must be continuously connected to the internet
- [D5] VAT code uniquely identifies a third party in the system
- [D6] Third parties know the monitored user Fiscal Code
- [D7] Third parties are able to provide callback interfaces
- [D8] AutomatedSOS has access to parameters that can be useful for identifying an emergency status
- [D9] There exists an API through which AutomatedSOS can pass to an ambulance information about the identity, the location and the vital parameters of the person in trouble
- [D10] Track4Run athlete wears the monitoring device during the whole run
- [D11] There is a system providing an API to visualize GPS data on a map.

### **2.4.2 Software dependencies**

The considered system relies on external software interfaces to accomplish the previously described functions and make the resulting product more lightweight.

- **Ambulance Calling System**

AutomatedSOS must call an ambulance to rescue the user given his geographical position. For the sake of simplicity, we assume that the national health service offers to the external applications an API that automatically calls an ambulance and sends to it the coordinates to reach the mentioned location.

- **City Maps**

Track4Run users need to use city maps to accomplish different tasks: organizers use them to create events and trace the path of the runs, athletes to check the information about the track in the created events and spectators to display the real-time positions of the athletes on the path during the run. An API of this kind could be Google Maps API, which gives accurate real-time information for mapping, navigation, and places.

- **Mobile Operating System**

The two subsystems, AutomatedSOS and Track4Run, are mobile applications that run on some physical smart-devices. For this reason, they need an operating system to interface with such as Android or iOS.

### **2.4.3 Hardware constraints**

- **Servers**

Our system needs to store huge amounts of data coming from single users. For this reason, a strong underlying server architecture is needed to develop the back-end part of the system. Possibly the resulting architecture can take inspiration from the classic three-tier web application architecture composed of a web server, an application server, and a database server.

- **Sensors and Smart Devices**

Data4Help acquires data from the previously defined data sources, which are devices such as smart-watches, smartphones, wearables and other. Hence, even if they are not directly part of the system, the reliability and availability of the system's data are strongly connected to the existence of reliable sensors and devices that can collect and send data from them in real-time.

- **GPS**

As concerns the geographical position of the users, Data4Help should give it to the subscribed third parties, if requested. Moreover, AutomatedSOS must know it to send correct info to the ambulances and Track4Run use the user's geographical position to display it on the run path to the spectators during the run. For this reason, in order to retrieve the correct value for the geographical position of the user, this last should have at least one active device with GPS enabled.

- **2G/3G/4G internet connection**

Even if the user is not connected via WiFi or Ethernet cable to the internet, our system (in particular AutomatedSOS and Track4Run) has to retrieve user's data in spite of the user's current position. For this reason, the system relies on 3G / 4G technology to grant to the user a non-stop internet access.

## 3 Specific Requirements

### 3.1 External Interface Requirements

In the following section we provide a description of the software interfaces offered to the external systems and users in order to access its functionalities. This includes user interface mockups as well as a definition of the APIs that will be realized for the third parties.

#### 3.1.1 User Interface

A subset of the possible UIs offered by our platform is here provided. As always we divide the interfaces according to the system under examination:

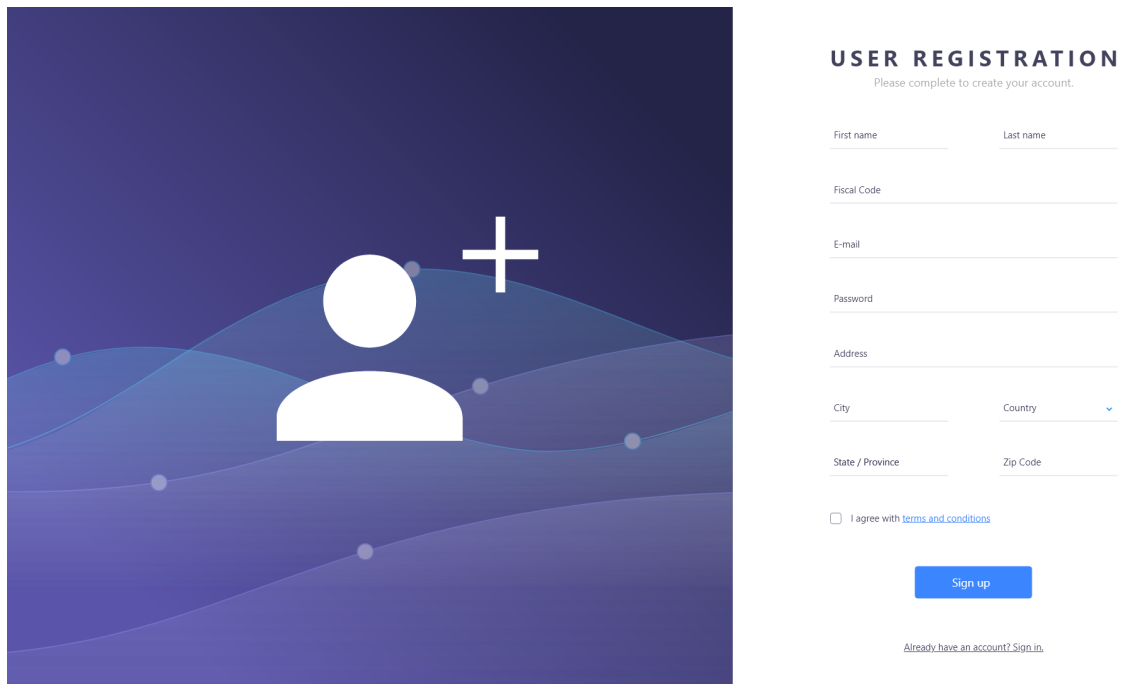
- **Data4Help**

For the users Data4Help is designed to be a web application: the structure of each page should be intuitive and user friendly, so that even the less experienced user in web browsing can easily use the system. On the other hand, a web application for the third parties doesn't exist. A parallel API is also offered to the third parties, which is described in later sections. For this reason there are no mockups for the third party UI of Data4Help.

- **AutomatedSOS and Track4Run**

These two subsystems are designed to be mobile applications. The following mockups show examples of the typical situations in which the user could find himself after installing the application and using it on his smart-phone. Hereunder mockups for other smart-devices are not included, but the user interfaces will be very similar to these ones.

An example of UI mockups can be found here below.



The image shows a user registration form on the right and a decorative graphic on the left. The graphic features a dark blue background with lighter blue wavy lines and a white silhouette of a person with a plus sign above their head. The form is titled "USER REGISTRATION" and includes a sub-header "Please complete to create your account." It contains several input fields: "First name", "Last name", "Fiscal Code", "E-mail", "Password", "Address", "City", "Country" (a dropdown menu), "State / Province", and "Zip Code". There is a checkbox for "I agree with [terms and conditions](#)". A blue "Sign up" button is located below the form. At the bottom, there is a link: "Already have an account? [Sign in.](#)"

### USER REGISTRATION

Please complete to create your account.

First name  Last name

Fiscal Code

E-mail

Password

Address

City  Country

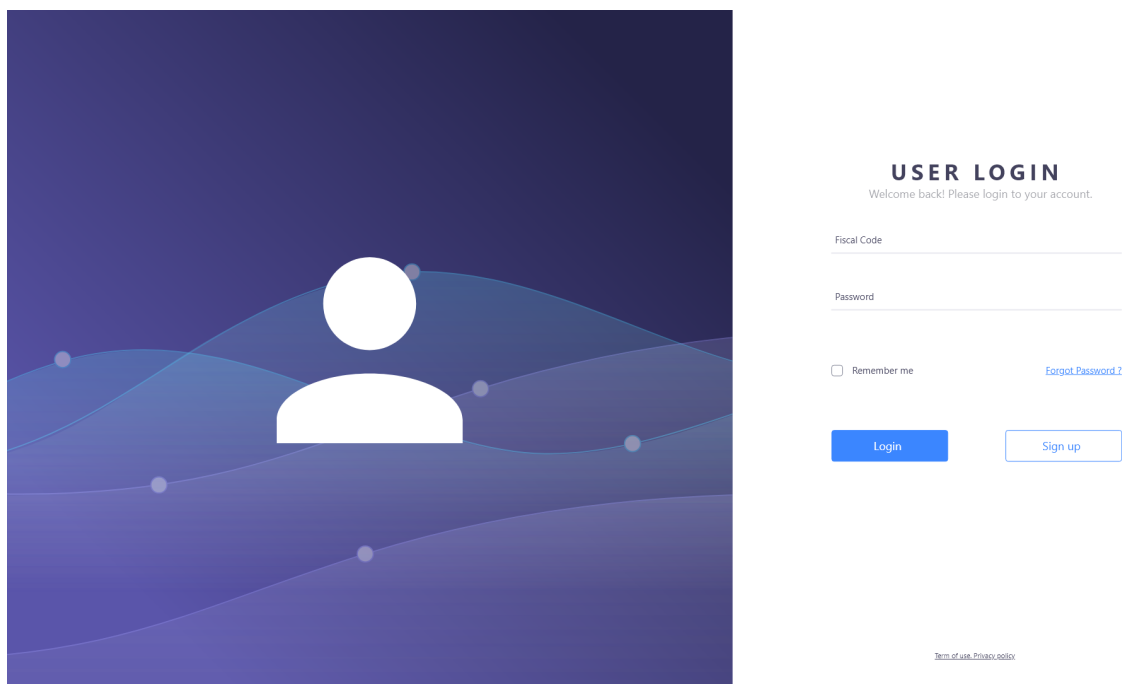
State / Province  Zip Code

☐ I agree with [terms and conditions](#)

[Sign up](#)

[Already have an account? Sign in.](#)

Figure 5: Data4Help - User Registration



The image shows a user login form on the right and a decorative graphic on the left. The graphic is identical to the one in Figure 5. The form is titled "USER LOGIN" and includes a sub-header "Welcome back! Please login to your account." It contains two input fields: "Fiscal Code" and "Password". There is a checkbox for "Remember me" and a link "Forgot Password?". Two buttons, "Login" and "Sign up", are located below the form. At the bottom, there is a link: "[Term of use](#) [Privacy policy](#)".

### USER LOGIN

Welcome back! Please login to your account.

Fiscal Code

Password

☐ Remember me [Forgot Password ?](#)

[Login](#) [Sign up](#)

[Term of use](#) [Privacy policy](#)

Figure 6: Data4Help - User Login



DATA4HELP

Home

Data Sources

Third Parties

Help Center

Settings

Q Enter a keyword ...

John Doe

Physical Devices

Enabled	Device	Position	Heart Rate	Pressure	Temperature	Glucose	Accellerometer	
ON	iPhone	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
ON	iClock	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
ON	nuFIBand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
OFF	GlucosCheck	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

External Applications

Enabled	Application	Position	Heart Rate	Pressure	Temperature	Glucose	Accellerometer	
ON	Eppol Health Care	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

✓ Save

Figure 7: Data4Help - User Data Sources Management

DATA4HELP

Home

Data Sources

Third Parties

Help Center

Settings

Q Enter a keyword ...

John Doe

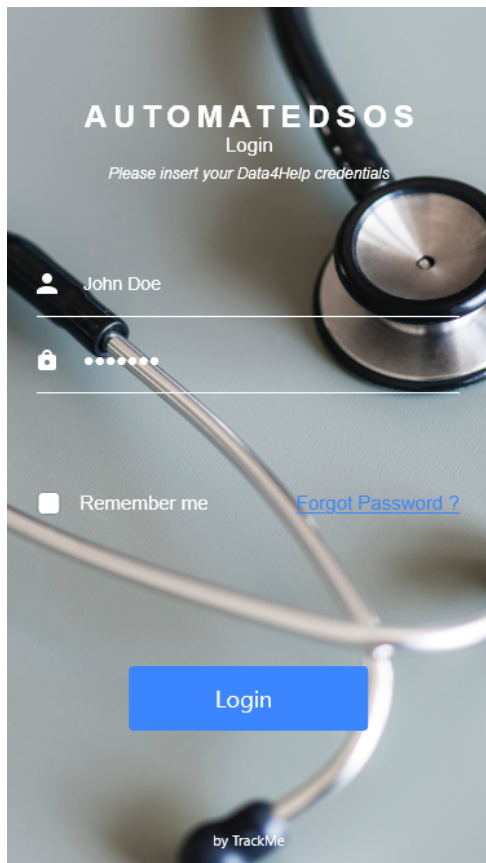
Subscribed Third Parties

Third Party	Subscription Date	Expiration Date	Revoke Permission
NanoSoft	10 / 10 / 2018	12 / 12 / 2018	
Boyer Pharmaceuticals	19 / 09 / 2018	20 / 12 / 2018	

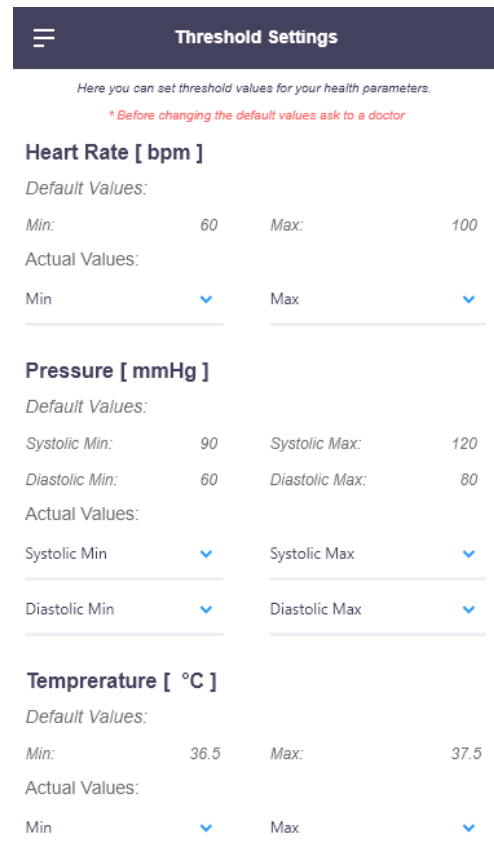
Pending Requests

Third Party	Subscription Period	Monitored Parameters		
Pluzer	25 / 11 / 2018 - 03 / 02 / 2019	Heart Rate, Pressure, Temperature	<div>✓ Accept</div>	<div>✗ Refuse</div>
VirgoActive	-	Heart Rate	<div>✓ Accept</div>	<div>✗ Refuse</div>
GSKU	-	Heart Rate, Pressure, Temperature	<div>✓ Accept</div>	<div>✗ Refuse</div>

Figure 8: Data4Help - User Third Parties Management



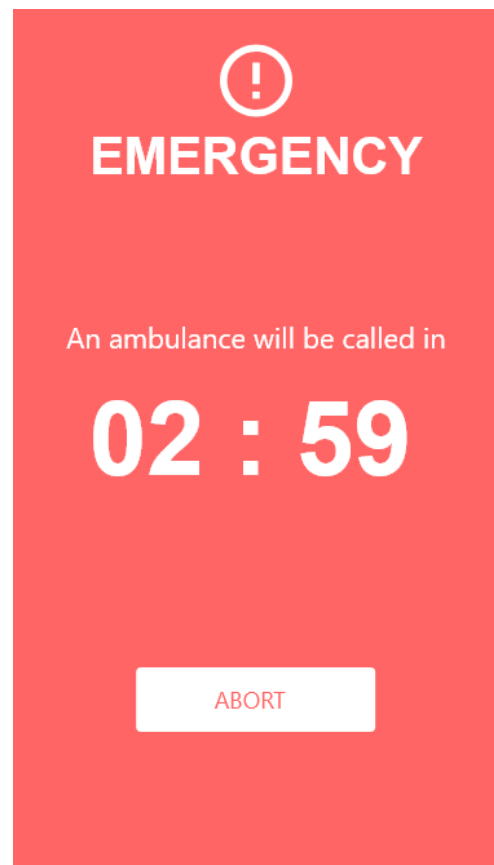
(a) Login



(b) Threshold Settings

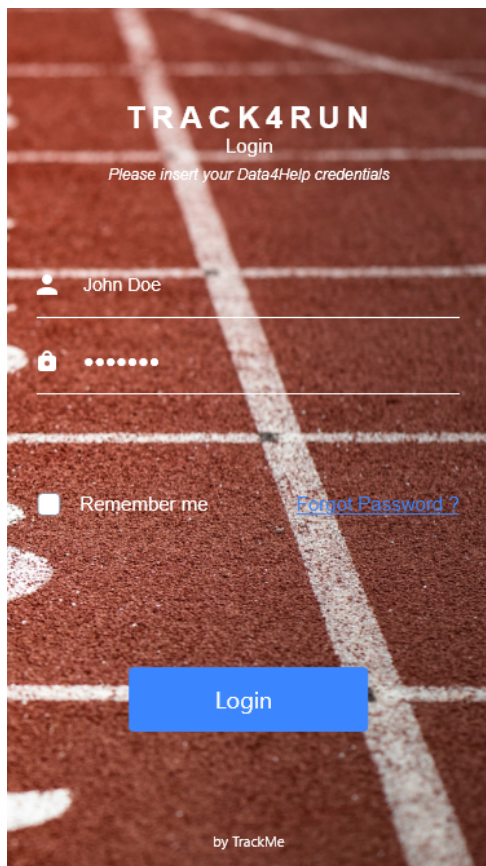


(c) myData

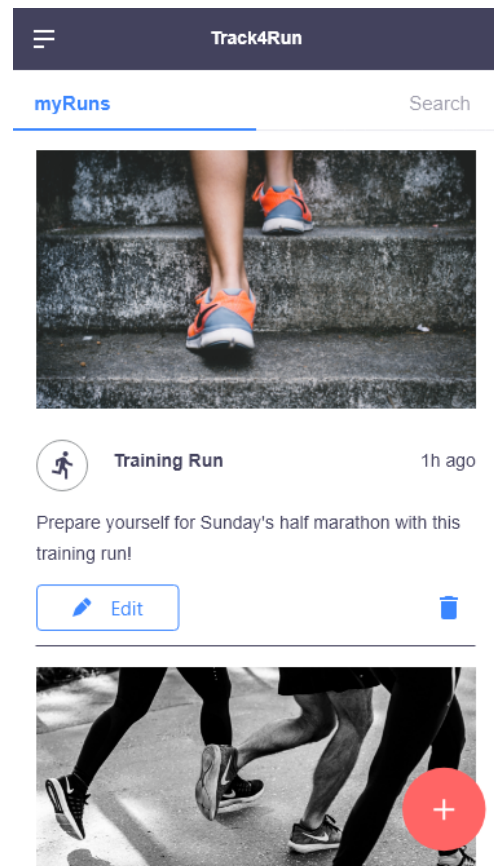


(d) Emergency Alert

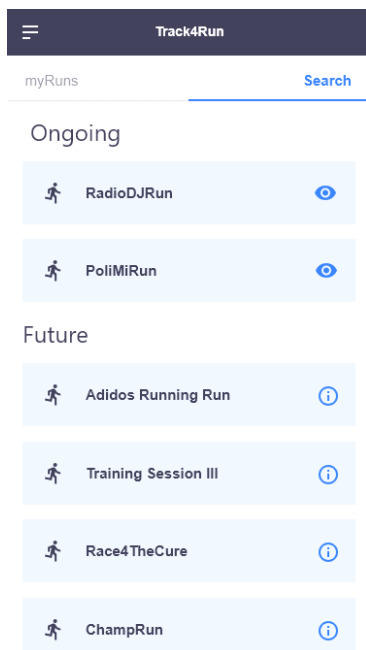
Figure 9: AutomatedSOS Mockups



(a) Login



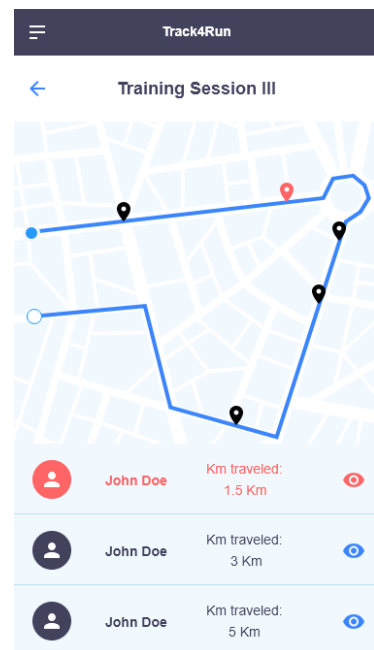
(b) myRuns



(c) Search



(d) Run Details



(e) Live Run Details

Figure 10: Track4Run Mockups

### 3.1.2 Hardware Interfaces

Since our system is designed to be a software platform running on any compatible hardware, no proprietary hardware interface is provided.

### 3.1.3 Software Interfaces

The main external software interfaces of our system can be found in Data4Help, which as already explained is the part responsible of connecting data producers to data consumers. To achieve this goal, Data4Help exposes two main software interfaces, which can be thought as APIs that are given to the third party applications in order to connect their software to ours.

- **Data Sharing Interface:** used by third parties to access users' data collected by the system. This interface shall offer four main functionalities: authentication, group data requests, single data requests, subscription requests.

Authentication must be done using a trusted way of identifying the organization that wants to register to the system, in order to prevent malicious actors to infiltrate the system. For example, this can be done using an electronic signature coupled with the VAT code of the company.

Group and single user data requests are defined by some filters that the third party wants to apply in order to retrieve only the needed data. When this is possible, they will generate a response containing the latest available data from the system.

Finally, subscription to new data can be performed by providing a callback interface, that will then be notified with some specific call whenever some new data is available on the system.

- **Data Acquisition Interface:** provided to the applications and hardware devices which collect data from the user, and gives the possibility to add this kind of information to the user's data set in the system.

As for third parties, authentication is necessary in this case too. This can be provided by the data source using the user's Data4Help credentials, so that the systems knows which user is adding that data source.

Once authenticated, the data source is provided with an interface to send data streams to the Data4Help system, that will collect and store them safely.

### 3.1.4 Communication Interfaces

As communication interfaces, this system will use the commonly available protocols to access the internet and to make remote calls to software interfaces of external and internal systems.

## 3.2 Functional Requirements

### 3.2.1 Requirements

**[G1] Data4Help must be able to keep track of real time health status and position from registered users' devices**

- **[R1]** A User shall be able to register to TrackMe's services providing a Fiscal code and a password of their choice
- **[R2]** A User shall be able to unsubscribe from the service at any moment
- **[D1]** Fiscal code uniquely identifies a user of the system
- **[R3]** After registration, a user must be able to log in by using his/her credentials

- [R4] The system must acquire user data only after he/she accepted the data acquisition policy
- [R5] The system shall acquire data from external applications that register themselves as data sources of a user
- [D2] Every user owns at least one device capable of retrieving correct real-time health parameters and location
- [D3] User devices must grant to the system access to the requested data
- [D4] User devices must be continuously connected to the internet

**[G2] Data4Help should allow third parties to gather information from a single user or from an anonymous group of users**

- [R6] Third parties must be able to register to Data4Help by providing a VAT code
- [D5] VAT code uniquely identifies a third party in the system
- [R7] The system shall provide to the registered third parties unique access tokens to use the services
- [R8] The system must grant access to a single user data only after his/her confirmation
- [D6] Third parties know the monitored user Fiscal Code
- [R9] The system must be capable of merging multiple user data and anonymize it
- [R10] The system shall grant access to merged data only if the number of individuals whose data satisfy the request is higher than 1000

**[G3] Data4Help should allow third parties to subscribe to new data and receive them as soon as they're produced**

- [R11] The system must store information about third parties' data subscriptions
- [R12] The system asks to the third parties who want to subscribe to new data to provide a callback interface
- [D7] Third parties are able to provide callback interfaces
- [R13] The system must notify every observing third party through their callbacks whenever an observed data changes
- [R14] The system must provide an interface to start and manage requests
- [R15] The system must give the possibility to third parties to unsubscribe from user data

**[G4] AutomatedSOS should be able to identify an health emergency when the user data are below/exceeding a certain threshold**

- [R16] AutomatedSOS users must be signed in with a Data4Help account
- [R17] AutomatedSOS can use data coming from Data4Help to monitor the health condition of its users
- ~~[R18] AutomatedSOS, when possible, uses data coming directly from the device sensors.~~
- [R19] When activated, AutomatedSOS provides some default values for parameters thresholds

- [R20] AutomatedSOS gives the user the possibility to set his/her own threshold
- [D8] AutomatedSOS has access to parameters that can be useful for identifying an emergency status
- [R21] AutomatedSOS shall be able to unsubscribe from the service at any time

**[G5] AutomatedSOS must notify the user and call an ambulance when it detects a health emergency**

- [D9] There exists an API through which AutomatedSOS can pass to an ambulance information about the identity, the location and the vital parameters of the person in trouble
- [R22] The system must notify the user before calling the ambulance
- [R23] The system gives the possibility to abort the operation within 5 seconds
- [R24] The system automatically calls the ambulance if the operation is not aborted

**[G6] Track4Run allows an organizer to create and manage a run.**

- [R25] Track4Run users must be signed in with a Data4Help account
- [R26] Track4Run allows organizers to create a "run event" providing the track, the date and a starting and ending time
- [R27] Track4Run allows organizers to modify the details of the created event
- [R28] Track4Run allows organizers to delete his created runs
- [R29] Track4Run gives the possibility to the organizers to extend editing permissions of an event to other users (co-organizers)
- [R30] Track4Run can show to an organizer a list of his past, ongoing and future events

**[G7] Track4Run allows a user to participate to an organized run**

- [D10] Track4Run athlete wears the monitoring device during the whole run
- [R31] Track4Run gives the user the possibility to look up for future organized runs and join them
- [R32] Track4Run gives the user the possibility to unsubscribe from a previously joined run at any moment
- [R33] Track4Run must keep track of the position of every participant of the run from the start until the end of the event

**[G8] Track4Run allows a spectator to track the position of the participants of the run**

- [R34] Track4Run gives the spectator the possibility to look up for ongoing runs and select the one to follow
- [R35] Track4Run shows to the spectator the position of all the participants of the selected run on a map
- [D11] There is system providing an API to visualize GPS data on a map.

### 3.2.2 Scenarios

#### Data4Help - Scenario 1

While talking, John tells his friend Matt about a new web application called Data4Help that helps you keeping track of your health status. Hence Matt, interested in checking his heart rate, decides to register to it. To do this he surfs to Data4Help web site and fills a form with his personal information, including his fiscal code, and he chooses a username and password that will be used as credentials for the login. After filling the form he clicks on the final checkbox to accept the terms and conditions and complete his registration. Thereafter Matt receives a notification saying he has registered successfully to Data4Help.

#### Data4Help - Scenario 2

Mary, a Data4Help customer, received for her birthday the new Eppol iClock. Knowing that Eppol's products support Data4Help data source synchronization, she decides to pair her new device with her Data4Help account. She searches for this feature in her iClock and enables it. Soon a confirmation request arrives via email and she confirms it. The device is added to the already available data sources. After that, Mary goes to the Data4Help website, logs in, browse to the "Data Sources" section and sees the new iClock in the available data sources. There she assigns the tracking of the "Hearth Rate" parameter to her smartwatch. With the same procedure, she assigns to her smartphone the tracking of the "Position" parameter. She's then satisfied with her preferences and saves them.

#### Data4Help - Scenario 3

Michael Garcia, Boyer Pharmaceutical CEO, heard about the Data4Help service and, in agreement with the Administrative Board, he decided to introduce it in the company. The day after, Lukas, the developer of the company visits Data4Help web site and, after reading the API documentation, crafts a registration request with all the necessary information included (company name, VAT code, electronic signature, ...) and sends it. After a few minutes Lukas receives a confirmation email saying he has successfully registered the company to the Data4Help service. Therefore, the company will be able to access Data4Help services dedicated to the third parties.

#### Data4Help - Scenario 4

Brian, patient of the Lenox Private Medical Center, was released yesterday. The clinic, registered to the Data4Help service, decides to monitor his health status to see if the treatment he was under reached the desired results. The doctor that was in charge of Brian, by using the software of the clinic integrated with the Data4Help API, sends a request to monitor Brian's health status. As soon as the doctor sends the request, Brian receives an email asking for his permission. He accepts the request by clicking on the link included in the email and a confirmation email saying that the user accepted the request is sent to the Medical Center. Therefore the requested information are sent back to the hospital server and the software developed using the Data4Help API parses the collected data in order to make them readable for the doctor.

#### Data4Help - Scenario 5

Pfuzer, a big pharmaceutical company registered to Data4Help service, needs to gather the heart rate data of all the young people under 30 years old in Italy for a market analysis aimed at evaluating the production of a new drug against heart disease. For this reason, Todd Chavez, the marketing manager of the company, by using the software of the company integrated with the Data4Help API, creates a new group data request. He inserts *Italy* and *<30* in the filters for the location and the age and then he sends the request. Unfortunately, the software immediately notifies Todd saying that the request cannot be satisfied because the cardinality of the target users is less than 1000. For this reason Todd decides to untighten the search filters and sends the request again. After that Todd receives the requested data.

#### AutomatedSOS - Scenario 1

Andrew decided to buy the new iClock as a gift for the birthday of his grandfather Leonard, 93 years old, who suffers from severe heart problems and lives alone. Since Leonard would like to be autonomous, his new device with AuotmatedSOS service active on it, grants him a safer life. One night Leonard wakes up with severe chest pains. The iClock immediately detects the "heart rate" parameter exceeded the maximum threshold and shows a noisy emergency alert, saying that an ambulance will be called if the

user doesn't abort in the following minute. Leonard is really sick and is not able to abort the operation. In few minutes an ambulance arrives and Leonard is immediately rescued.

#### **AutomatedSOS - Scenario 2**

William suffers from epilepsy. The attacks are not very frequent but they are so strong that a few months ago he fell and he got a nasty head injury. While looking for an automated solution to check on his conditions, he hears about AutomatedSOS and decides to activate the service provided by Data4Help. At first he sets the threshold of the tracked parameters with the help of his doctor and he adds the contacts of his parents. When the wristwatch detects repetitive shaking motion, it automatically sends the user's bluetooth-connected phone text and call alerts to the designated recipients. Within seconds, family members receive these alerts, which include the date, time and GPS location of the event.

#### **AutomatedSOS - Scenario 3**

Steve, a AutomatedSOS customer with diabete, needs to periodically check the glucose levels observed by his medical device connected with the AutomatedSOS application on his smartphone. For doing this, he starts the application and goes on the "myData" tab. A nice view appears, containing all the information about his monitored health parameters with a lot of colorful diagrams. Steve filters out the displayed content by selecting the desired observation period. He's then presented with all the data gathered for that period.

#### **Track4Run - Scenario 1**

Mario is a personal trainer and he is organizing a run for all his athletes. He is thinking about using the Track4Run Service to accomplish this task. So he download the Track4Run application and logs in with his Data4Help credentials. Once the application is installed and running, he goes under the "myRuns" section and taps on the big plus button. At this point a configuration frame shows up. Luke starts by filling out a form: he adds the event name, the travelled distance and the starting and ending time. Finally, by clicking on a map, he marks all the checkpoints of the run. After a quick check, Luke clicks on the "Create" button and a confirmation alert appears saying that the event has been successfully published in the news feed.

#### **Track4Run - Scenario 2**

Chris, runner and Data4Help customer, is looking for a run to join. By looking at the news feed of Track4Run on his smartphone he sees the event created by Mario. He taps on the event name and the full description of the event shows up. After reading the description and all the details he decides to join the run, hence he clicks on the "Participate" button. A confirmation alert shows up, saying that the event has been added to the attending events.

#### **Track4Run - Scenario 3**

Katy, Chris' wife and Data4Help customer, wants to go cheer his husband at the run. The day of the run she launches the Track4Run and searches for his husband run under the "search" tab. Katy selects the desired run and a nice map with a the GPS position of all the athletes opens up.



### 3.2.3 Use Cases

Name	User registration
Actor	User
Entry conditions	User surfs to the Data4Help web page
Events flow	<ol style="list-style-type: none"><li>1. Fills the registration form</li><li>2. Accepts the terms and conditions by clicking on the checkbox</li><li>3. Clicks on the <i>Sign Up</i> button</li><li>4. The system process the registration request and sends back a notification to the user</li></ol>
Exit conditions	Registration is successful and the user is informed via notification
Exceptions	<ul style="list-style-type: none"><li>• User is already registered</li><li>• There is some invalid data in the form</li><li>• The email is already used</li><li>• Terms and conditions haven't been checked</li></ul> <p>The exception is notified to the user and the registration procedure restart</p>

Name	User Login
Actor	User
Entry conditions	User is already registered and enters Data4Help web page
Events flow	<ol style="list-style-type: none"><li>1. The user enters his credentials</li><li>2. Clicks on the <i>Enter</i> button</li><li>3. The log in is successful and the user is redirected to the home page of the web application</li></ol>
Exit conditions	The log in is successful and the user is redirected to home page
Exceptions	<ul style="list-style-type: none"><li>• Credentials are not valid</li></ul> <p>The exception is notified to the user and the login procedure restart</p>

Name	Add Data Source
Actor	User
Entry conditions	User synchronizes his device or application with Data4Help
Events flow	<ol style="list-style-type: none"> <li>1. A confirmation email is sent to the user</li> <li>2. The user clicks on the confirmation link included in the email</li> <li>3. The data source is added to the list of the user data sources</li> </ol>
Exit conditions	The user confirms the synchronization
Exceptions	<ul style="list-style-type: none"> <li>• The user didn't request the synchronization and doesn't click on the confirmation</li> </ul> <p>After 24 hours the request is made void and a notification email is sent to the user</p>

Name	Data Sources Management
Actor	User
Entry conditions	The user enters the <i>Data Sources</i> section in the Data4Help web site
Events flow	<ol style="list-style-type: none"> <li>1. The list of data sources connected to the user account is shown</li> <li>2. The user selects which data source to configure</li> <li>3. For that data source, the list of all the possible parameters to track is shown</li> <li>4. The user turns On/Off the tracking of each parameter</li> <li>5. Clicks on the <i>Save</i> button</li> </ol>
Exit conditions	The user have set his preferences and saves them
Exceptions	<ul style="list-style-type: none"> <li>• The parameter is already tracked from a more reliable source</li> </ul>

Name	Third Party Registration
Actor	Third Party
Entry conditions	Third party has installed Data4Help software
Events flow	<ol style="list-style-type: none"> <li>1. Using the Data4Help software client, the third party crafts a registration request including the access credentials</li> <li>2. Sends the registration request</li> </ol>
Exit conditions	Registration is successful and the third party is informed via notification
Exceptions	<ul style="list-style-type: none"> <li>• The third party is already registered</li> <li>• Invalid credentials used for the registration</li> </ul> <p>All the exceptions are notified to the third party with a response saying the registration procedure failed</p>

Name	Third Party Login
Actor	Third Party
Entry conditions	Third party is registered to Data4Help
Events flow	<ol style="list-style-type: none"> <li>1. Using the Data4Help software client, the third party crafts a login request including the access credentials</li> <li>2. Sends the login request</li> </ol>
Exit conditions	Login is successful and the third party is informed via notification
Exceptions	<ul style="list-style-type: none"> <li>• Invalid credentials used for the login</li> </ul> <p>All the exceptions are notified to the third party with a response saying the login procedure failed</p>

Name	One Shot Single Data Request
Actor	Third Party, User
Entry conditions	Third party is registered to Data4Help
Events flow	<ol style="list-style-type: none"> <li>1. Using the Data4Help software client, the third party crafts a request including the fiscal code of the recipient, <i>one-shot</i> as the request type and the parameters of the target user to monitor</li> <li>2. Sends the request</li> <li>3. A notification is sent via email to the user</li> <li>4. The user clicks on the confirmation link included in the email</li> <li>5. The results are sent via email to the third party</li> </ol>
Exit conditions	The user has responded to the request and, if successful, the data are made available to the third party
Exceptions	<ul style="list-style-type: none"> <li>• A registered user with the specified fiscal code doesn't exist</li> <li>• The user doesn't accept the request</li> </ul> <p>All the exceptions are notified to the third party with a brief description of the problem</p>

Name	Subscription Single Data Request
Actor	Third Party, User
Entry conditions	Third party is registered to Data4Help
Events flow	<ol style="list-style-type: none"> <li>1. Using the Data4Help software client, the third party crafts a request including the fiscal code of the recipient, <i>subscription</i> as the request type, the subscription period and the parameters of the target user to monitor</li> <li>2. Sends the request</li> <li>3. A notification is sent via email to the user</li> <li>4. The user clicks on the confirmation link included in the email</li> <li>5. If the request is successful, the information are sent to the third party. Every time the requested data change, the new data are sent to the third party</li> </ol>
Exit conditions	The subscription period ends, user unsubscribes from the request
Exceptions	<ul style="list-style-type: none"> <li>• A registered user with the specified fiscal code doesn't exist</li> <li>• The user doesn't accept the request</li> </ul> <p>All the exceptions are notified to the third party with a brief description of the problem</p>

Name	One Shot Group Data Request
Actor	Third Party
Entry conditions	Third party is registered to Data4Help
Events flow	<ol style="list-style-type: none"> <li>1. Using the Data4Help software client, the third party crafts a request including the filters that specify the scope of the request(location, age, ...), <i>one-shot</i> as the request type and the parameters of the target users to monitor</li> <li>2. Sends the request</li> <li>3. If the request is successful, the information are sent to the third party</li> </ol>
Exit conditions	The request was successful and the data are made available to the third party
Exceptions	<ul style="list-style-type: none"> <li>• Request is rejected due to lack of anonymization (less than 1000 users)</li> </ul> <p>All the exceptions are notified to the third party with a brief description of the problem</p>

Name	Subscription Group Data Request
Actor	Third Party
Entry conditions	Third party is registered to Data4Help
Events flow	<ol style="list-style-type: none"> <li>1. Using the Data4Help software client, the third party crafts a request including the filters that specify the scope of the request(location, age, ...), <i>subscription</i> as the request type, the subscription period and the parameters of the target users to monitor</li> <li>2. Sends the request</li> <li>3. If the request is successful, the information are sent to the third party</li> <li>4. Every time the requested data change, the new data are sent to the third party</li> </ol>
Exit conditions	The subscription period ends
Exceptions	<ol style="list-style-type: none"> <li>1. Request is rejected due to lack of anonymization (less than 1000 users satisfy the search filters)</li> </ol> <p>All the exceptions are notified to the third party with a brief description of the problem</p>

Name	AutomatedSOS Login
Actor	User
Entry conditions	The user previously installed the AutomatedSOS application on his smart-phone
Events flow	<ol style="list-style-type: none"> <li>1. The user starts the application</li> <li>2. Enters his Data4Help credentials</li> <li>3. Clicks on the <i>Login</i> button</li> </ol>
Exit conditions	Login successful
Exceptions	<ul style="list-style-type: none"> <li>• Credentials are not valid</li> </ul> <p>The exceptions are notified to the user and the login procedure restarts</p>

Name	Monitoring Service
Actor	AutomatedSOS, User, Ambulance
Entry conditions	The user is logged in the AutomatedSOS application
Events flow	<ol style="list-style-type: none"> <li>1. The application continuously checks if the data coming from user's data sources satisfy threshold constraints</li> <li>2. Whenever a parameter is found below or above its thresholds the user is given an emergency status</li> <li>3. On the user's smartdevice, an emergency status screen with a countdown to call an ambulance shows up</li> <li>4. An API call for an ambulance in the users' with emergency status location is made</li> </ol>
Exit conditions	User unsubscribes from Data4Help service or uninstalls AutomatedSOS application
Exceptions	<ul style="list-style-type: none"> <li>• Lost physical device for data acquisition</li> </ul> <p>This exception is notified to the client and pause the Monitoring System until the problem is fixed</p> <ul style="list-style-type: none"> <li>• User clicks on the <i>Abort</i> button in the emergency status screen</li> </ul> <p>This exception revokes the emergency status of the user</p>

Name	Thresholds Setting
Actor	User
Entry conditions	User enters in the "Thresholds Setting" tab of the AutomatedSOS application
Events flow	<ol style="list-style-type: none"> <li>1. The application shows to the user the tracked parameters, the default values and the saved values for each of them</li> <li>2. For each parameter the user can change the current values of the thresholds by choosing from a dropdown</li> <li>3. The user saves the changes</li> </ol>
Exit conditions	The user saves and exits
Exceptions	

Name	MyData
Actor	User
Entry conditions	User enters in the "MyData" tab of the AutomatedSOS application
Events flow	<ol style="list-style-type: none"> <li>1. The application shows to the user all the gathered info</li> <li>2. The user can select from a dropdown the observation period</li> <li>3. Whenever a filter is changed the application responds with the filtered information</li> </ol>
Exit conditions	The user exits the "myData" tab
Exceptions	<ul style="list-style-type: none"> <li>• The system hasn't gathered any data yet</li> </ul> <p>The exceptions are notified to the user and the "myData" page is empty</p>

Name	Create a Run
Actor	User (Organizer)
Entry conditions	The organizer has previously installed the Track4Run application
Events flow	<ol style="list-style-type: none"> <li>1. The organizer clicks on the + icon on the bottom of the "myRuns" section</li> <li>2. Enters the information about the run</li> <li>3. Selects the path of the run using a map</li> <li>4. Clicks on the <i>Create</i> button</li> </ol>
Exit conditions	The organizer has successfully created a run
Exceptions	



Name	Run enroll
Actor	User (Runner)
Entry conditions	The runner has previously installed the Track4Run application
Events flow	<ol style="list-style-type: none"> <li>1. The runner clicks on the "Search" tab</li> <li>2. Choose from the "Future" list of runs which one to enroll</li> <li>3. Clicks on the "Enroll" button</li> </ol>
Exit conditions	The user has successfully enrolled to the run
Exceptions	<ol style="list-style-type: none"> <li>1. The user doesn't possess a device able to Track the runner</li> <li>2. The user is in a bad shape for a run, medical advice is suggested</li> </ol> <p>This exceptions is notified to the client and the procedure goes on</p>

Name	Run unsubscription
Actor	User (Runner)
Entry conditions	User already enrolled on a Run and he/she want to unsubscribe
Events flow	<ol style="list-style-type: none"> <li>1. Looks up under "My Runs" tab for the previously joined run</li> <li>2. Clicks on the "Unsubscribe" button</li> </ol>
Exit conditions	User has successfully unsubscribed from the run
Exceptions	

Name	Watch Run
Actor	User (Spectator), Users (Runners)
Entry conditions	A spectator goes under the "Search" section of Track4Run
Events flow	<ol style="list-style-type: none"> <li>1. Select a Run form the list of ongoing runs</li> <li>2. Watch the map of the run with the real time GPS tracking of the runners</li> </ol>
Exit conditions	Run ends or spectator exits from the application
Exceptions	<ul style="list-style-type: none"> <li>• Runner has connection problems</li> <li>• Spectator has connection problems</li> <li>• Server has connection problems</li> </ul> <p>All exception are notified to the spectator and the process goes on</p>

Name	Run modification
Actor	User (Organizer)
Entry conditions	The organizer goes under the "myRuns" tab of the Track4Run application
Events flow	<ol style="list-style-type: none"> <li>1. Select the run you want to modify from the one presented in the timeline</li> <li>2. Clicks on the "Edit" button</li> <li>3. Modify the information of the run</li> <li>4. Click on the "Confirm" button</li> </ol>
Exit conditions	The organizer successfully modified the run and the participants are notified
Exceptions	

Name	Run deletion
Actor	User (Organizer)
Entry conditions	The organizer goes under the "myRuns" tab of the Track4Run application
Events flow	<ol style="list-style-type: none"> <li>1. Select the run you want to modify from the one presented in the timeline</li> <li>2. Clicks on the trash bin icon</li> </ol>
Exit conditions	The organizer successfully deleted the run and the participants are notified
Exceptions	

### 3.2.4 Sequence Diagrams

#### Data4Help Sequence Diagrams

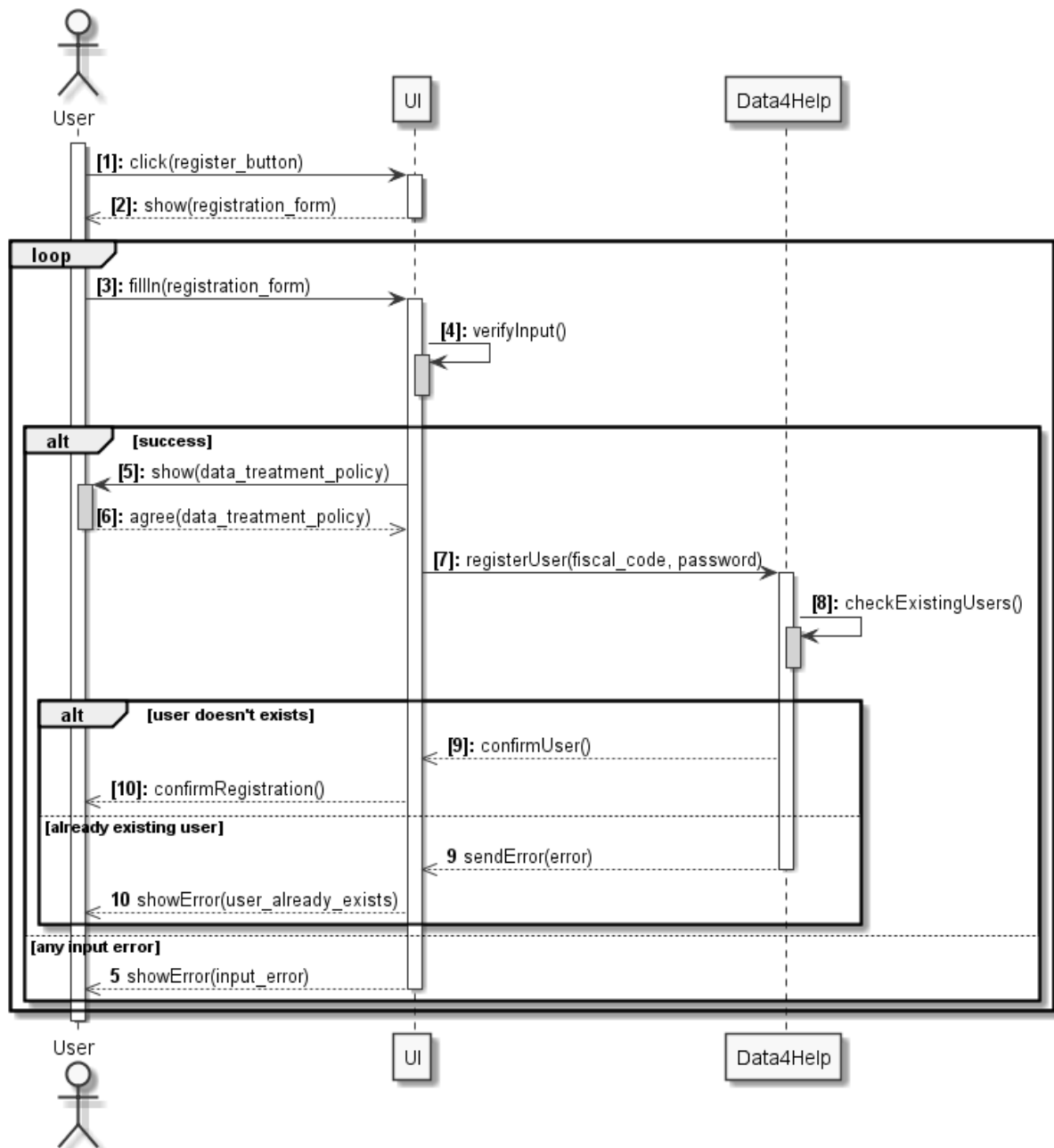


Figure 11: Data4Help - User Registration Sequence

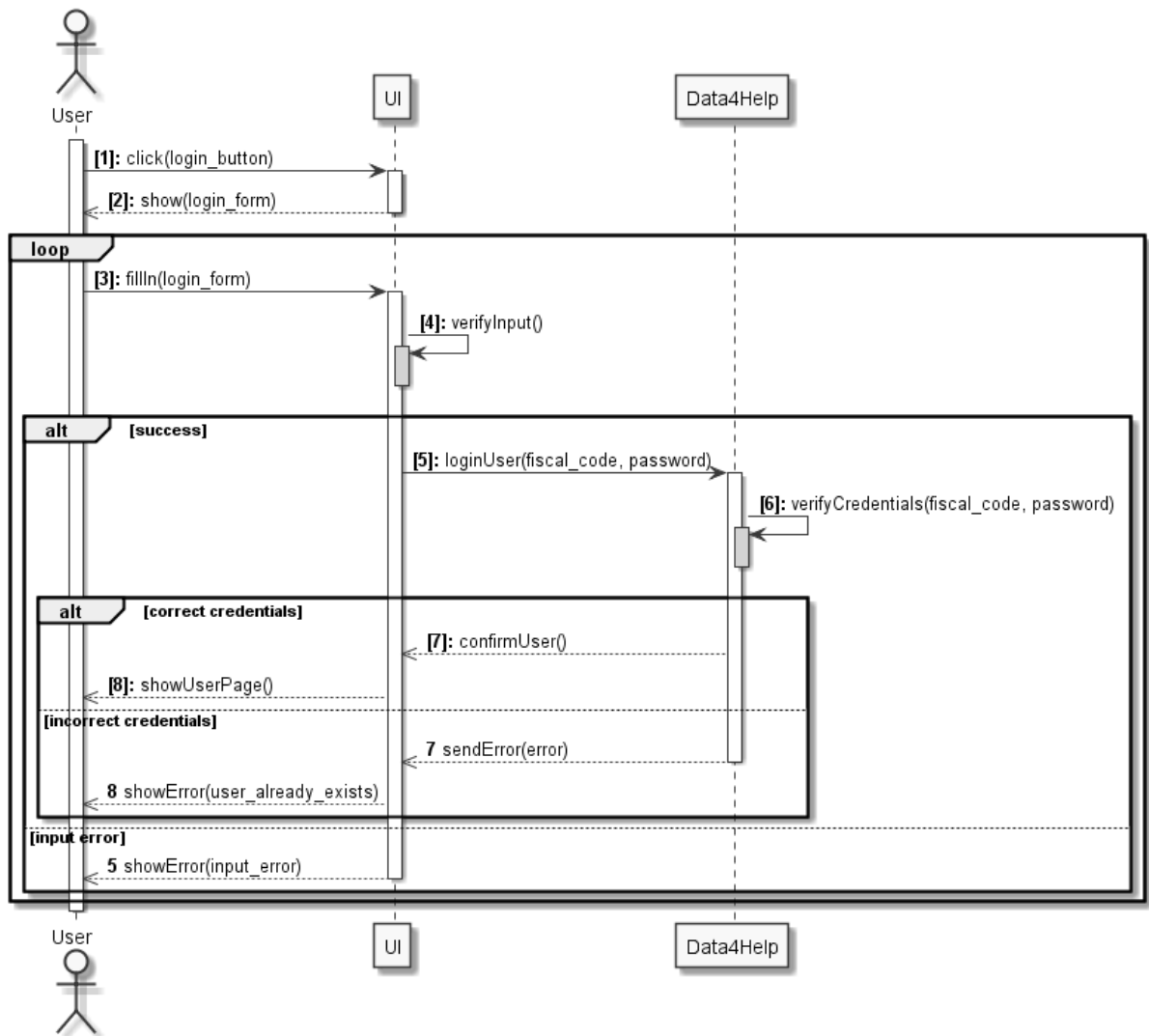


Figure 12: Data4Help - User Login Sequence

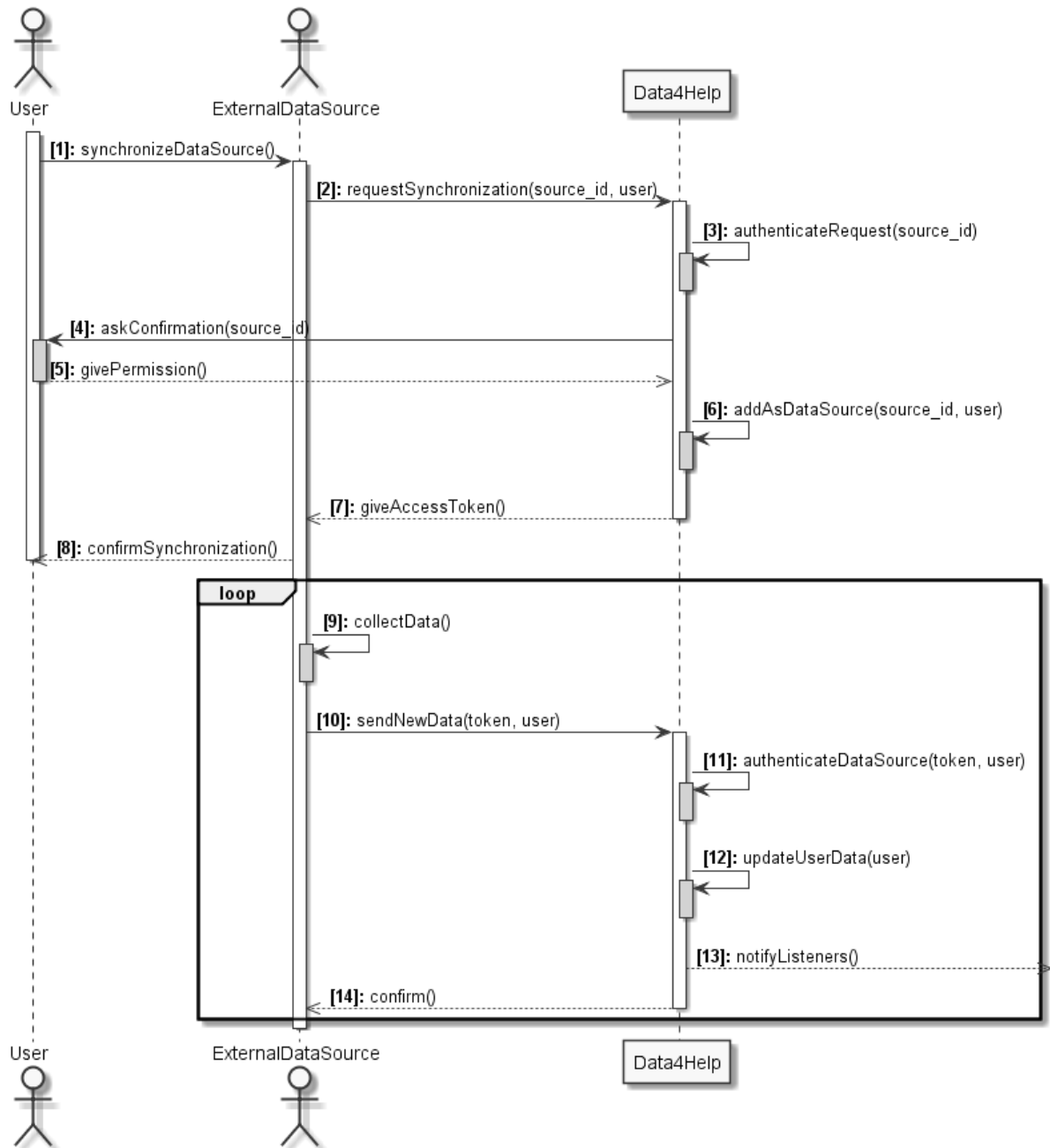


Figure 13: Data4Help - Data Source Synchronization Sequence

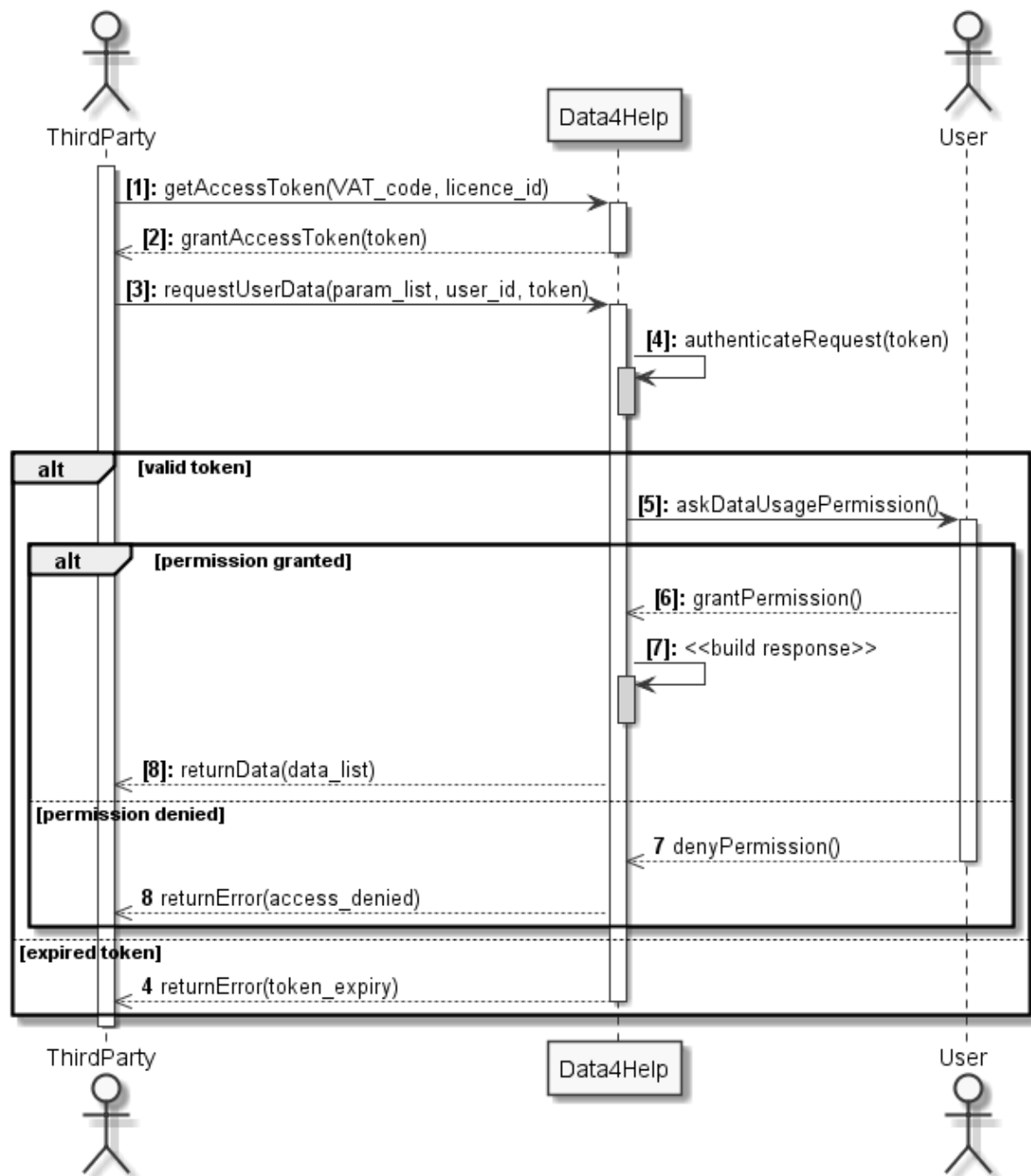


Figure 14: Data4Help - Third Party single data request sequence

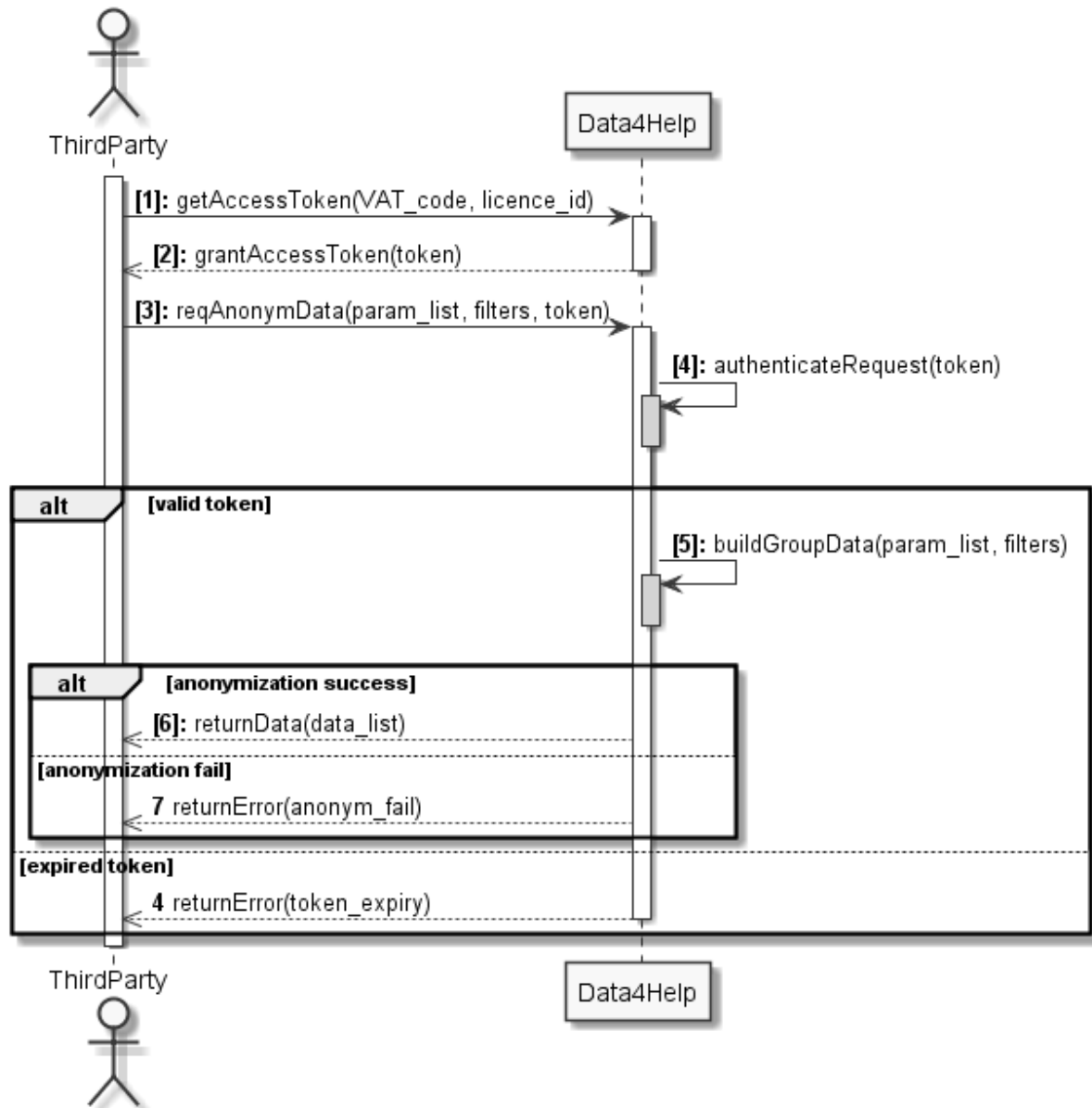


Figure 15: Data4Help - Third Party group data request sequence

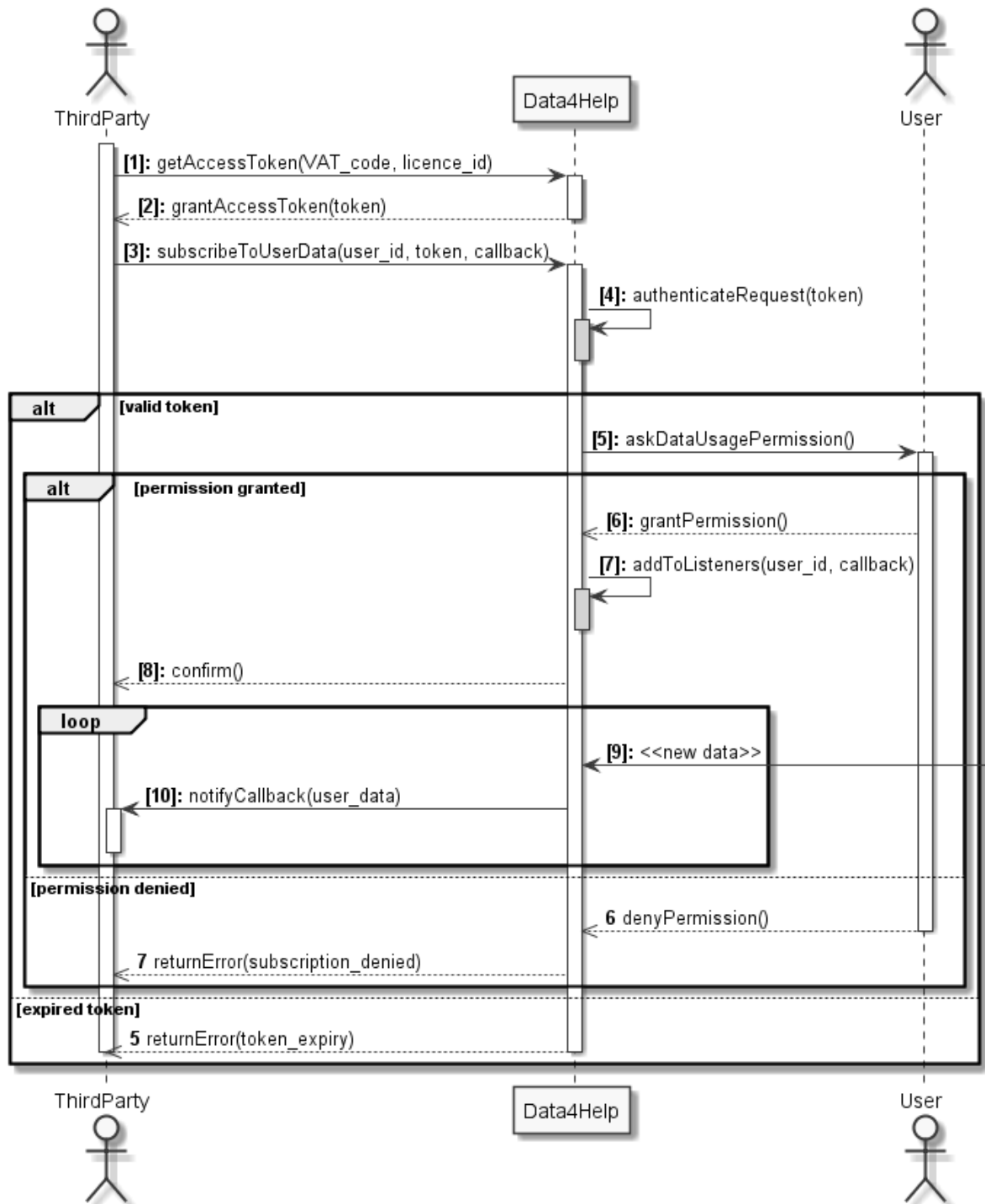


Figure 16: Data4Help - Third Party subscription request sequence



## AutomatedSOS Sequence Diagrams

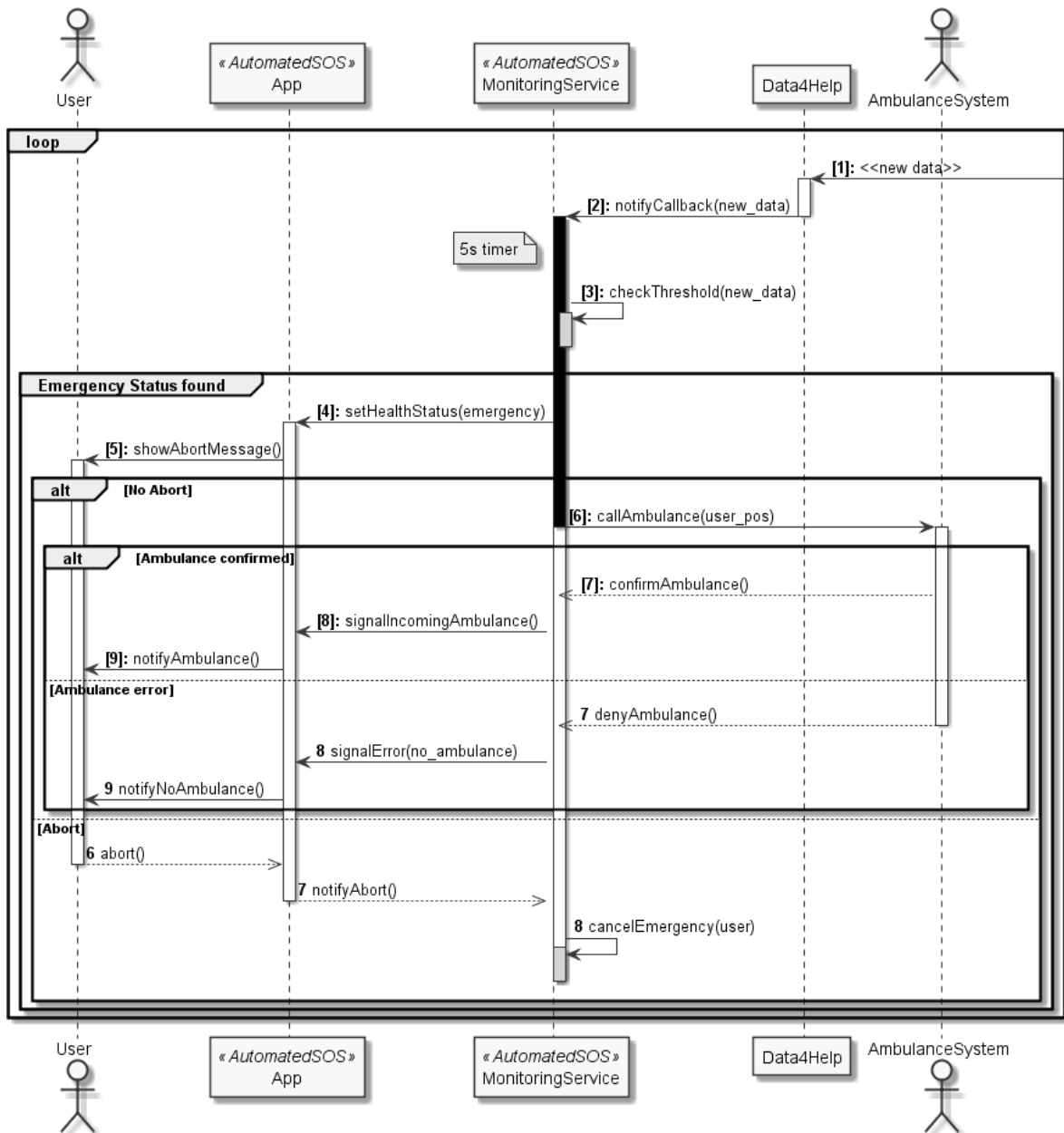


Figure 17: AutomatedSOS - Emergency sequence

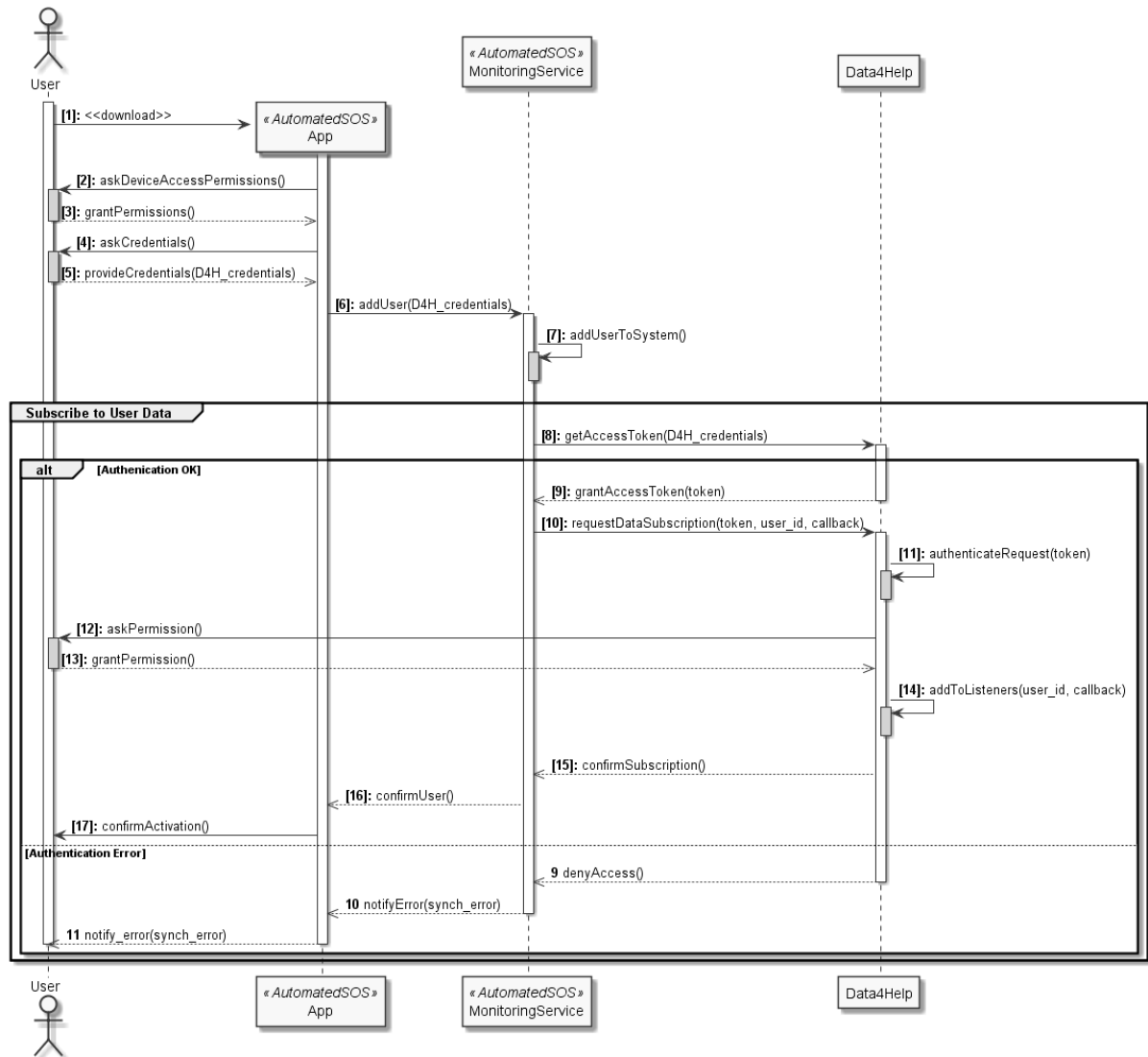


Figure 18: AutomatedSOS - User Registration sequence

## Track4Run Sequence Diagrams

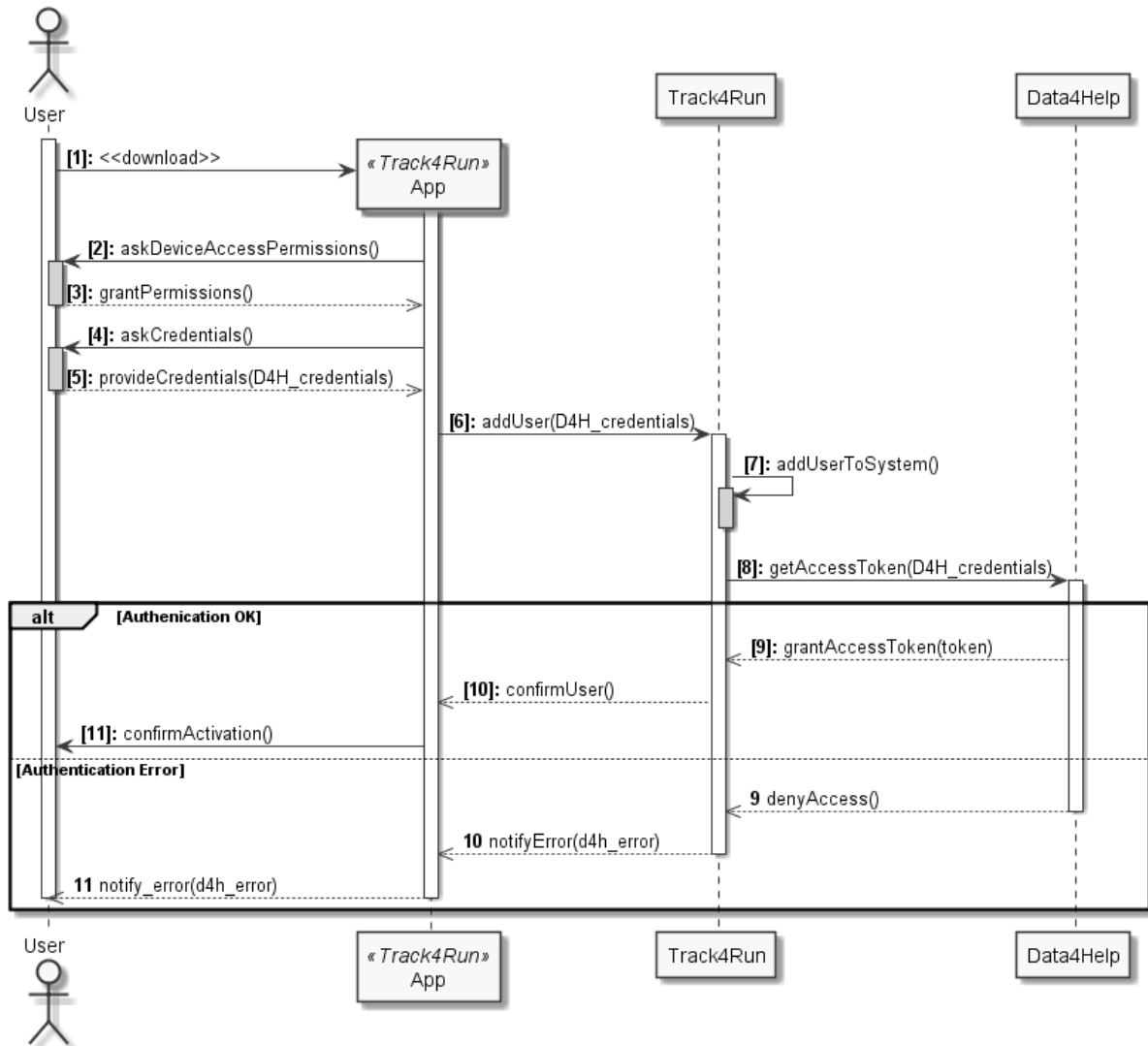


Figure 19: Track4Run - User Registration sequence

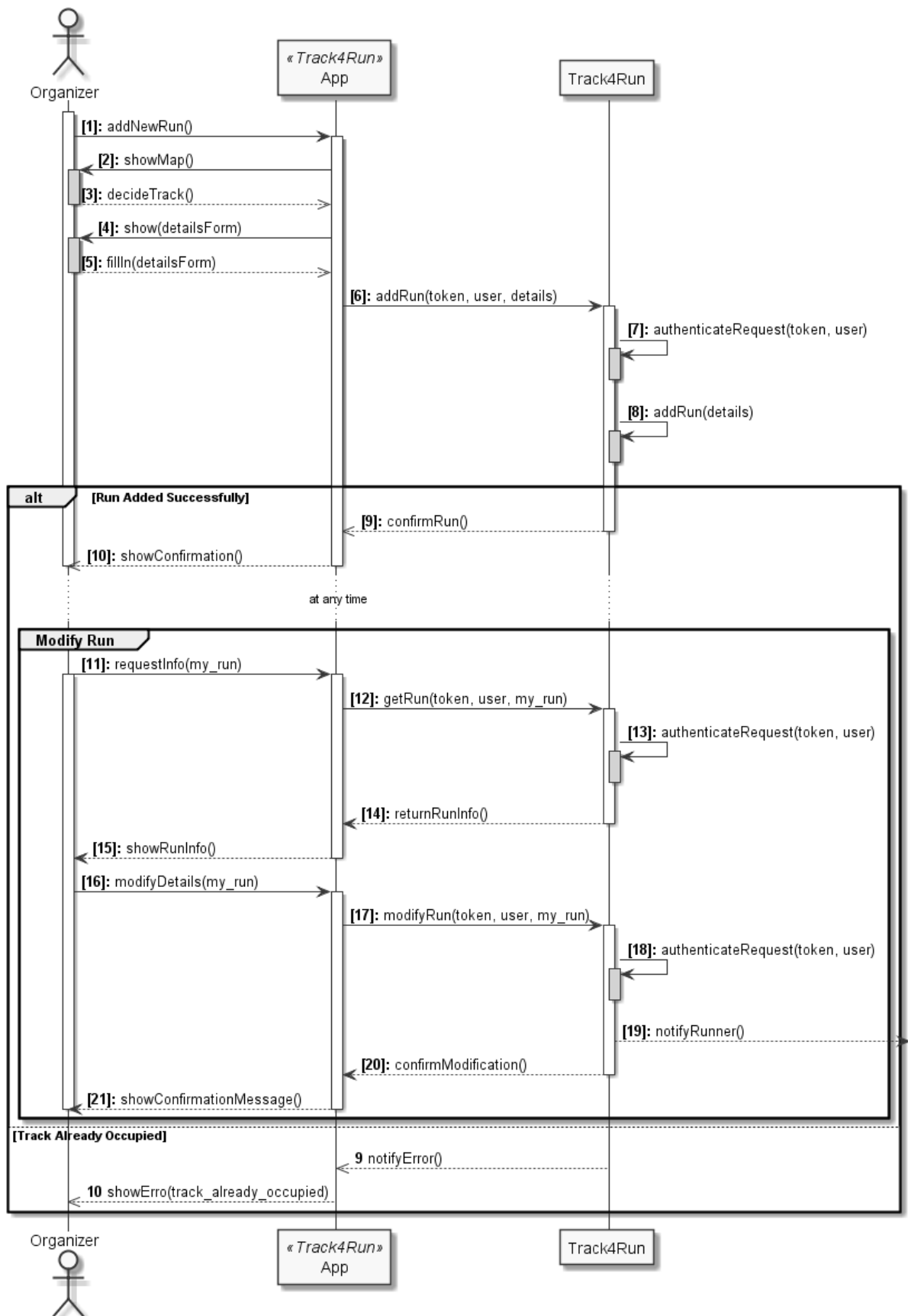


Figure 20: Track4Run - Run Creation sequence

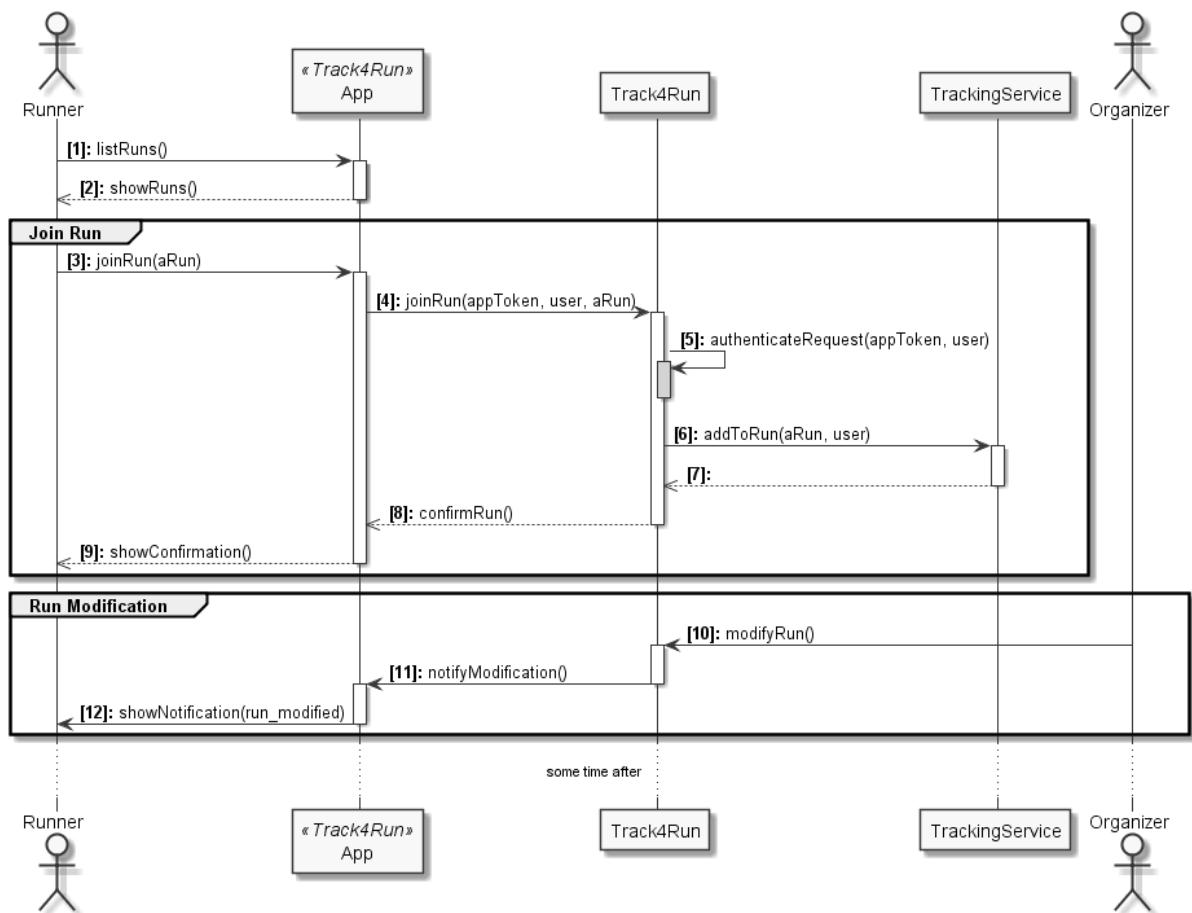


Figure 21: Track4Run - Run joining sequence

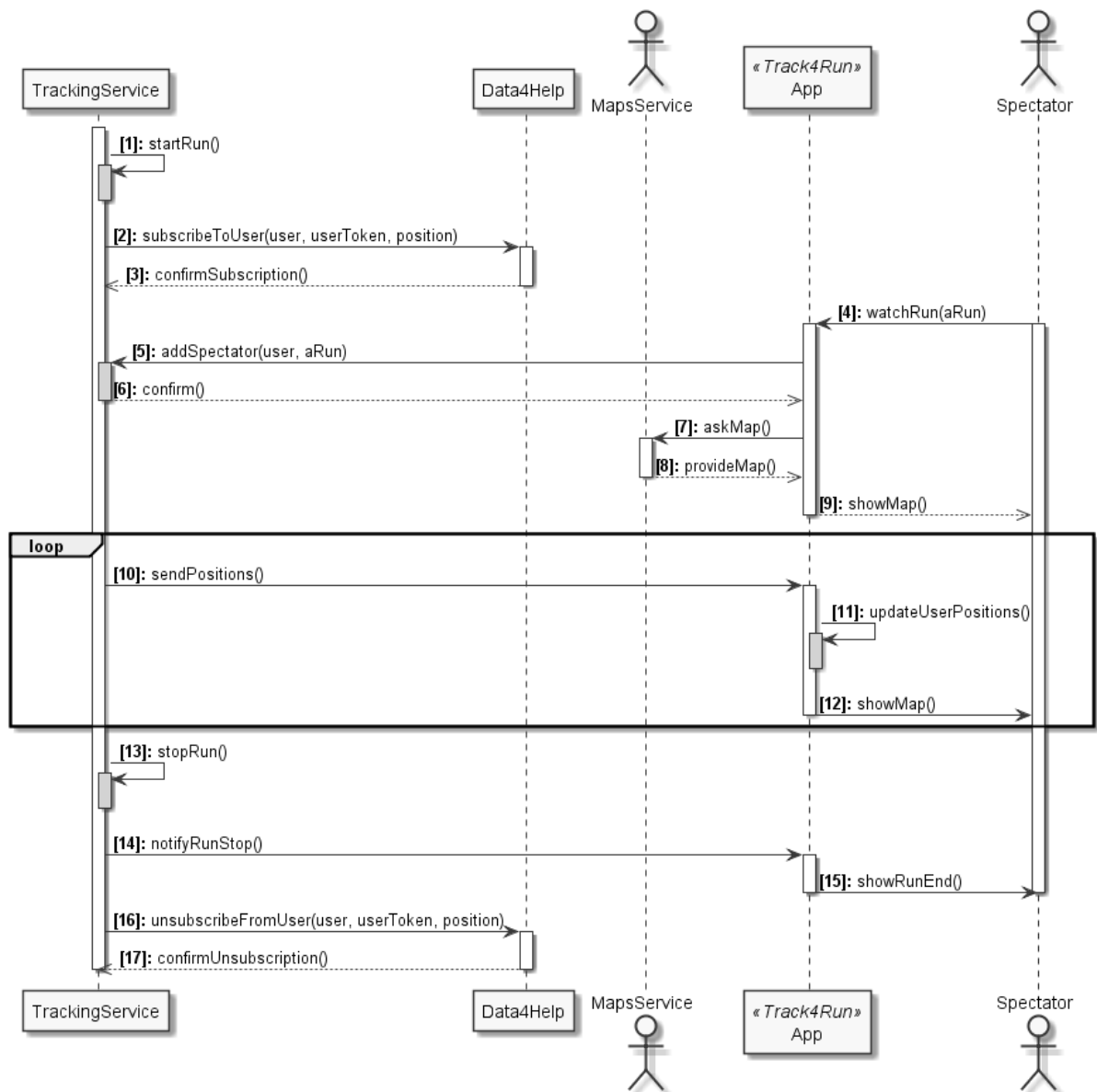


Figure 22: Track4Run - Spectator watching run sequence

### 3.3 Performance Requirements

In general, the main performance requirements for our system concern the time needed to process data and fulfill data requests.

In particular, the system should meet the following strong requirement:

- **AutomatedSOS Emergency Response** The system must call an ambulance within **5 seconds** from when a parameter exceeds a threshold. This time shall take into account the time needed for the AutomatedSOS monitoring system to process new incoming data and detect the emergency, the Data4Help system to send the new data to AutomatedSOS and the average delay introduced from the communication interfaces.

### 3.4 Design Constraints

#### 3.4.1 Regulatory policies

- **GDPR:** In order to protect the user's privacy and perform a correct data treatment, the system must be developed in compliance with the latest GDPR regulations, giving the maximum attention to the protection and anonymization of the user's data.

#### 3.4.2 Hardware limitations

The only hardware limitations of our system is present in the devices on which AutomatedSOS and Track4Run applications will be installed. For this reason, the development of the software to be will have to reasonably take in account some common limitations of these devices, such as power consumption and memory size.

### 3.5 Software System Attributes

#### 3.5.1 Reliability

The system must be available 24/7. Since this requirement is quite demanding, small service breaks will be tolerated. To boost reliability, a RAID architecture that combines multiple disk drives to have data redundancy is suggested. Using a RAID controller we could also expand disks' capacity or substitute them without the risk of losing data or suspending the service for too long. Preventive maintenance is also a good idea to avoid downtimes.

#### 3.5.2 Availability

To guarantee a 3-nines (99.9%) availability degree, a system of redundant servers could be considered. In this way we don't have a single point of failure and if one server fails, another one will be ready to substitute it.

#### 3.5.3 Security

Security is a main issue. At first, to access the functionalities offered by our platform, users have to complete a login phase providing their credentials: fiscal code and password in case of a single user or VAT code and electronic signature for the third parties. These sensitive information should be confidentially stored and encrypted using a proper hash function.

Secondly, the system continuously collects data on health conditions from the users. Typically this kind of data are also considered sensitive and should be kept secret. Finally, also communications between users and our platform is very important, for this reason the system could be accessed by the customers using secure connection protocols like HTTPS to avoid Man In The Middle attacks. Only the

system administrator who is responsible of the Web and Application Server configuration can decide if it is the case to implement them or not.

#### **3.5.4 Maintainability**

In order to make our software the most maintainable as possible, we will adopt a modular design for our code. In this different components of the system can be modified and additional ones can be introduced in the system with whenever is needed and with a minimal impact on the rest of the system. Moreover, we will make sure to adopt standard design patterns and coding best practices, so that our code can be easily understood and modified by any future developer. Finally, the code should be completely and properly documented in order to facilitate the understanding of it.

#### **3.5.5 Portability**

During the system installation and setup, we have to consider different factors:

- Ease of installation on the central server
- Scalability of the platform considering future adjustments without losing the already collected data
- Portability of data between different machines, in order to move the system on several more powerful machines in case of necessity up 24/7



## 4 Formal Analysis Using Alloy

In this final section there is the alloy model, the world generated and the proof of consistency. The model mainly describes the relationships between the various Data4Help entities. It's mainly shown how:

- It's impossible that multiple data sources of the same user tracks the same parameter.
- Requests with filters (not targeted at a single user's data, but at a group of users' data defined by that filters and anonymized) are accepted by the system only if the filters cover a wide enough group to guarantee privacy.
- Each user of AutomatedSOS is monitored by the system and, if hearth beat is lower or higher than the threshold range, the user is signaled to be in danger.
- Each user that want to enroll in a run is monitored on his position by Track4Run
- Each Spectator of Track4Run can only watch ongoing run and only if in that run are involved runners which positions are tracked

Moreover the world generated has 2 projected signatures: DataSources and Int. The consistency between the threshold values is nonetheless valid, even if omitted for readability. The same reasoning applies for DataSources, their are omitted to keep the image readable.

```
/** Signatures */
open util/boolean

sig User {
  parameters: some TrackedParameter,
  sources: some DataSource
}
sig DataSource {
  owner: one User,
  parameters: some TrackedParameter
}

sig TrackedParameter {
  source: one DataSource,
  user: one User,
  type: one ParameterType,
  value: one Int,
  threshold: lone Threshold
}

abstract sig ParameterType {}
one sig Position extends ParameterType {}
one sig HeartRate extends ParameterType {}
one sig Pressure extends ParameterType {}

sig ThirdParty {
  requests: set Request
}
one sig AutomatedSOS extends ThirdParty {}

sig Request {
  owner: one ThirdParty,
  status: one Status,
  filters: set Filter,
  targetUser: lone User
} { ((filters ≠ none) iff (targetUser = none)) and ((filters = none)
iff (targetUser ≠ none)) }

sig Filter {
  ownerRequest: one Request,
  paramType: one ParameterType,
```

```

min: one Int,
max: one Int
}

sig Group {
members : set User,
request: one Request
}

abstract sig Status {}
one sig Accepted extends Status {}
one sig Denied extends Status {}

sig Threshold {
min: one Int,
max: one Int,
parameter: one TrackedParameter
}

abstract sig EmergencyStatus {}
one sig Normal extends EmergencyStatus {}
one sig Emergency extends EmergencyStatus {}

sig AutomatedSOSUser extends User {
status: one EmergencyStatus
}

sig Runner extends User {
participate: some Run
}
sig Organizer extends User {
organize: some Run
}
sig Spectator extends User {
watching: one Run
}
sig Run {
owner: one Organizer,
participants: set Runner,
maxParticipants: one Int,
ongoing: one Bool
} {#participants ≤ maxParticipants}

one sig Track4Run extends ThirdParty {}

/** Data4Help Facts */

/* User Condition*/
-- Consistency of signatures connections
fact UserHasParam {
all u: User , p: TrackedParameter | (p.user = u iff (p in u.parameters ))
}
fact UserSourcesOwnership {
all u: User , s: DataSource | ( (s in u.sources) implies s.owner = u ) and ( s.owner = u
↪ implies ( s in u.sources ) )
}
fact SourceParamConnection {
all s: DataSource, p: TrackedParameter | ( (p in s.parameters) implies p.source = s) and
↪ ( p.source = s implies (p in s.parameters))
}
fact UserSourceTrackedParamConnection {
all u: User | all s: DataSource | all p: TrackedParameter |
(( s in u.sources) and (p in s.parameters)) implies (p in u.parameters)
}
-- For each Parameter Type a declaration that every User has at max one tracked instance
↪ of it
fact userMaxOneParamPerType {
all u: User | no disj tp1,tp2: TrackedParameter | ((tp1 in u.parameters) and (tp2 in u.
↪ parameters)) and ( tp1.type = tp2.type)
}
-- No filter of the same request can affect the same parameter type
fact filterMaxOneParamPerType {

```

```

no disj f1,f2: Filter | (f1.ownerRequest = f2.ownerRequest) and (f1.paramType = f2.
    ↪ paramType)
}

/* ThirdParty & Requests */
fact ThirdPartyRequestOwnership {
all tp: ThirdParty | all r: Request | (r in tp.requests) iff ( tp = r.owner)
}
fact RequestFilterOwnership {
all r: Request | all f: Filter | (f in r.filters) iff ( r = f.ownerRequest)
}
-- Each Request for a targetUser is as default accepted (the acceptance is the one made
    ↪ by the system, prior to user real acceptance of a request)
fact targetUserRequestAreAccepted {
all r: Request | (r.targetUser ≠ none) implies (r.status = Accepted)
}
-- for the sake of simplicity and because the scope is limited, the threshold for
    ↪ anonymization is set to 2
fact filteredRequestAcceptance {
all r: Request, g: Group | (r in g.request) implies ( ( r.status = Accepted) iff ( #g.
    ↪ members ≥2) )
}
fact RequestAcceptance {
all r: Request, g: Group | (r.status in Accepted) iff ( not(r.targetUser = none) or ((r
    ↪ in g.request) and (#g.members ≥ 2)))
}

/* Groups */
-- A request refers to a group of user <=> it has filters and it doesn't refers to a
    ↪ single target user
fact GroupsBelongToFilteredRequests {
all g:Group, r:Request | (r = g.request) iff ((r.filters ≠ none) and (r.targetUser = none
    ↪ ))
}

-- If a request refers to a group of user and the filters are on a ParamType present in
    ↪ that group user's TrackedParameters, than the TrackedParam's value are within the
    ↪ filter range
fact GroupMembersAreFiltered {
all g:Group | all u:User | all r:Request | all f:Filter | all p:TrackedParameter |      (
    ↪ u in g.members) and (r = g.request) and (f in r.filters) and (p in u.parameters)
    ↪ implies ((f.paramType = p.type) implies (p.value ≥ f.min and p.value ≤ f.max))
}
-- A request without filter doesn't refer to a group of user, while a request with filter
    ↪ refers at max to one group of user
fact eachFilterReqHasGroup {
all r:Request | ((r.filters = none) implies ((r.targetUser ≠ none) and (no g:Group | g.
    ↪ request = r)))
ãÑÑ
    and ((r.filters ≠ none) implies ((r.targetUser = none) and (no
    ↪ disj g1,g2:Group | g1.request = g2.request)))
}
-- for each user in a group of a request, exactly one TrackedParam of that user has type
    ↪ equal to a filter of that group
fact GroupMembersHaveFilteredParams {
all g:Group, u:User, r:Request, f:Filter |
ãÑÑ
    ((u in g.members) and (r = g.request) and (f in r.filters))
    ↪ implies (one p:TrackedParameter | (p in u.parameters) and (p.type = f.paramType))
}

/** AutomatedSOS */

-- AutomatedSOS makes request only for AutomatedSOS Users
fact AutoSOSMonitorOnlyAutoSOSUser {
all r: Request | (r.owner = AutomatedSOS) implies (one au: AutomatedSOSUser |(r.
    ↪ targetUser = au))
}

-- Threshold ownerParam and TrackedParameter threshold must be consistent
fact thresholdTrackedParamConsistency {
all th: Threshold, p: TrackedParameter | (p.threshold = th) iff (th.parameter = p)
}
-- AutomatedSOS makes one request for each AutomatedSOS User

```

```

fact allAutoSOSUserAreMonitored {
all au: AutomatedSOSUser | one r: Request | (r.owner = AutomatedSOS) and (r.targetUser =
    ↪ au)
}
-- All AutomatedSOSUser have HearthRate tracked
fact AutomatedSOSTrackedParamPresence {
all au: AutomatedSOSUser | one p: TrackedParameter | (p in au.parameters) and (p.type =
    ↪ HearthRate)
}
-- All AutomatedSOSUser have a threshold set for their HearthRate
fact thresholdPresence {
all au: AutomatedSOSUser, p: TrackedParameter | ((p in au.parameters) and (p.type =
    ↪ HearthRate)) implies (one th: Threshold | th in p.threshold)
}
-- Only AutomatedSOSUser have a threshold set for any TrackedParameter, any other User
    ↪ has no threshold
fact thresholdAbsenceForNonAutoSOSUser {
all u: User | not(u in AutomatedSOSUser) implies (no p: TrackedParameter | (p in u.
    ↪ parameters) and (one th: Threshold | th in p.threshold))
}
/* TrackedParameters != HearthRate cannot have threshold
   ÆÆ (this is valid only for this model, if other parameters needs to be checked in
    ↪ order to control user's health status this and other facts need to change)*/
fact OnlyHearthRateIsMonitored {
all p: TrackedParameter | (p.type ≠ HearthRate) implies (no th: Threshold | (th in p.
    ↪ threshold))
}
-- A AutomatedSOSUser is in an emergency status <=> his HearthRate broke the Threshold
fact userNeedsAmbulance {
all au: AutomatedSOSUser | (one p: TrackedParameter | (p in au.parameters) and (p.type =
    ↪ HearthRate) and (p.value > p.threshold.max or p.value < p.threshold.min)) iff (au.
    ↪ status = Emergency)
}

/** Track4Run */
-- Consistency of Track4Run signatures
fact OrganizerRunConsistency {
all r: Run, o: Organizer | (r in o.organize) iff (o = r.owner)
}
fact RunnerRunConsistency {
all rn: Runner, r: Run | (rn in r.participants) iff (r in rn.participate)
}
-- All Runners iscribed to Track4Run must have a device able to track their position
fact runnerPositionPresence {
all rn: Runner | one p: TrackedParameter | (p in rn.parameters) and (p.type = Position)
}
-- Track4Run makes request only to Runners
fact track4RunMonitorsOnlyRunners {
all r: Request | (r.owner = Track4Run) implies (one rn: Runner | (r.targetUser = rn))
}
-- Track4Run makes one request for each Runner
fact allRunnersAreMonitored {
all rn: Runner | one r: Request | (r.owner = Track4Run) and (r.targetUser = rn)
}
-- All Runner have Position tracked
fact runnerPositionPresence {
all rn: Runner | one p: TrackedParameter | (p in rn.parameters) and (p.type = Position)
}

pred runIsFull[r: Run] {
#r.participants = r.maxParticipants
}
pred enrollToRun[rn: Runner, r: Run] {
not runIsFull[r]
r.participants = r.participants + rn
}
pred watchRun[r: Run, s: Spectator] {
r.ongoing = True
some rn: Runner | (rn in r.participants) and (one p: TrackedParameter | (p in rn.
    ↪ parameters) and (p.type = Position))
s.watching = r
}

```

```

}

/* Int conversion to positive values + min/max consistency */
fact intToPositiveAndConsistency {
all tp: TrackedParameter, f: Filter, th: Threshold | (tp.value ≥ 0)
âĀĀĀ    and (f.min ≥ 0) and (f.max ≥ 0) and (f.min < f.max)
âĀĀĀ    and (th.min ≥ 0) and (th.max ≥ 0) and (th.min < th.max)
}

-- Check that every request that violates the anonymization restriction ( at least 2 user
  ↳ respect the filters ) is denied by the system
assert GroupDataAcceptance {
all r: Request, g: Group | (r.status in Denied) iff ((r.targetUser = none) and ((r in g.
  ↳ request) and ( #g.members < 2)))
}

-- Check that every AutomatedSOSUser that has his HearthBeat in the safe zone of its
  ↳ threshold can't be in an emergency status
assert AutoSOSUserHealthyAreNotInEmergency {
all au: AutomatedSOSUser | all p: TrackedParameter | ((p in au.parameters) and (p.type =
  ↳ HearthRate) and (p.value > p.threshold.min and p.value < p.threshold.max)) implies
  ↳ (au.status = Normal)
}

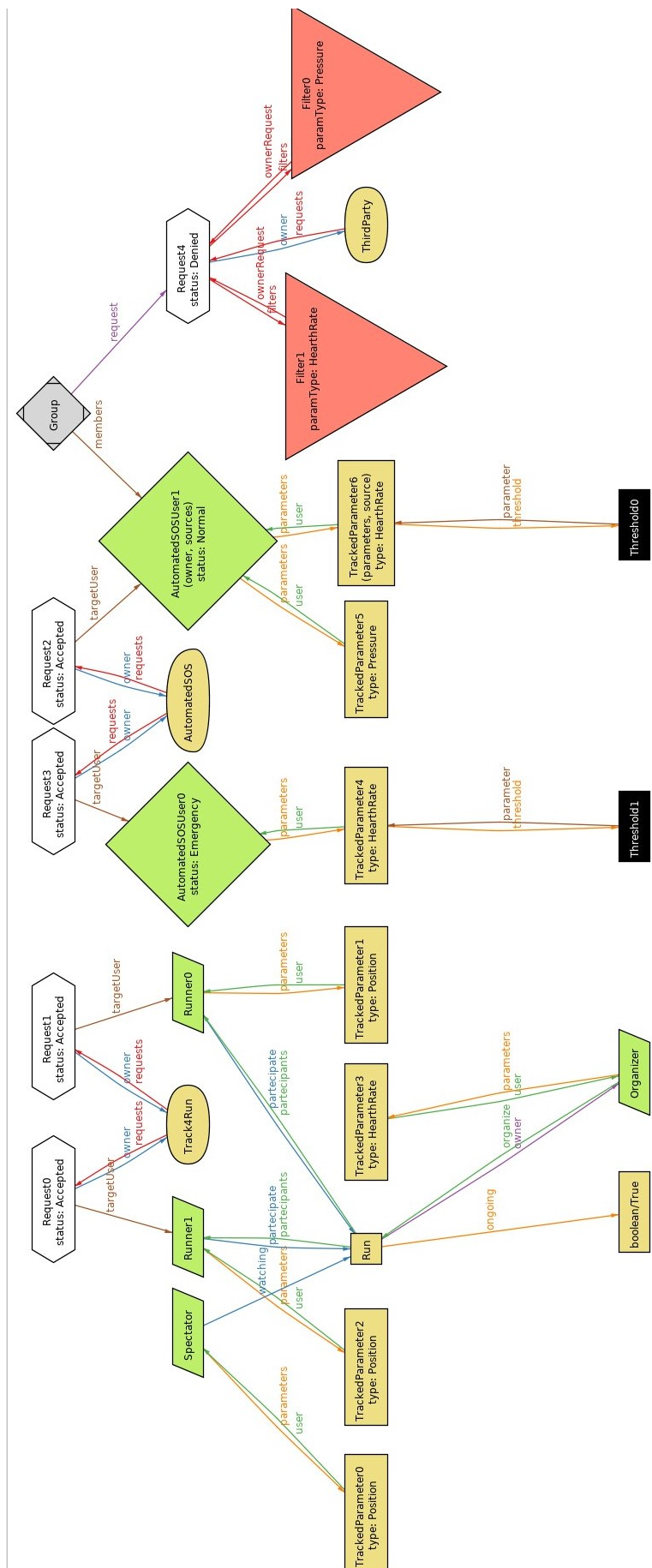
pred show {
(#AutomatedSOSUser = 2)
(one au: AutomatedSOSUser | au.status = Normal)
(one au: AutomatedSOSUser | au.status = Emergency)
(#Runner = 2)
(#Spectator = 1)
(#Run = 1)
( #Group ≥ 1)
( #Request ≥ 1)
(one g: Group | #g.members = 1) //This is to show at least one Request that is declined
( lone u: User | #u.parameters = 3) //Only one user will have all 3 parameters, to keep
  ↳ the World Generated Readable
}

run show for 8 but exactly 6 User, exactly 5 Request, exactly 3 ThirdParty, exactly 2
  ↳ Filter
run enrollToRun for 8
run watchRun for 8
check GroupDataAcceptance for 8
check AutoSOSUserHealthyAreNotInEmergency for 8

```



## 4.1 World Generated



## 4.2 Proof of consistency

### Executing "Run show for 8 but exactly 6 User, exactly 5 Request, exactly 3 ThirdParty, exactly 2 Filter"

Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20  
45881 vars. 1564 primary vars. 96312 clauses. 362ms.

**Instance found.** Predicate is consistent. 217ms.

### Executing "Run enrollToRun for 8"

Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20  
177319 vars. 2262 primary vars. 279909 clauses. 2967ms.

**Instance found.** Predicate is consistent. 309ms.

### Executing "Run watchRun for 8"

Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20  
177372 vars. 2270 primary vars. 279923 clauses. 2995ms.

**Instance found.** Predicate is consistent. 595ms.

### Executing "Check GroupDataAcceptance for 8"

Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20  
177304 vars. 2262 primary vars. 280035 clauses. 2994ms.

No counterexample found. Assertion may be valid. 58ms.

### Executing "Check AutoSOSUserHealthyAreNotInEmergency for 8"

Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20  
178094 vars. 2262 primary vars. 282647 clauses. 3024ms.

No counterexample found. Assertion may be valid. 615ms.

### 5 commands were executed. The results are:

- #1: **Instance found.** show is consistent.
- #2: **Instance found.** enrollToRun is consistent.
- #3: **Instance found.** watchRun is consistent.
- #4: No counterexample found. GroupDataAcceptance may be valid.
- #5: No counterexample found. AutoSOSUserHealthyAreNotInEmergency may be valid.



## **5 Effort Spent**

### **5.1 Andrea Biscontini**

- General brainstorming : 5h
- Requirements brainstorming: 7h
- Alloy model : 10h
- Final review : 15h
- Alloy review : 8h
- Scenarios and Use Case review : 1h

### **5.2 Alvise de' Faveri Tron**

- General brainstorming : 5h
- Requirements brainstorming: 7h
- UML models : 8h
- Final review : 15h
- Alloy review : 2h

### **5.3 Marco Gelli**

- General brainstorming : 5h
- Requirements brainstorming: 7h
- UI mockups : 9h
- Final review : 15h
- Scenarios and Use Case review : 5h

## 6 Reference

### 6.1 Reference

- Assessment: *Mandatory Project Assignment AY 2018-2019.pdf*
- [ISO/IEC/IEEE 29148 - Dec 2011](#)
- [Alloy Documentation](#)
- [Alloy cheatsheet from Iowa university](#)

### 6.2 Tools Used

- [Overleaf](#) - online  $\text{\LaTeX}$  editor
- [TexMaker](#) -  $\text{\LaTeX}$  editor
- [Alloy Analyzer](#)
- [Draw.io](#) for the UML diagrams
- [PlantUML](#) for the Sequence Diagrams